UNIT 1

DEFINITION OF MACHINE LEARNING AND ITS TYPES

Structure

1.1 Introduction
Expected Learning Outcomes
1.2 Definition of Machine Learning and its Brief Historical Overview
1.3 Measurement Scales in Statistics, Machine Learning and R Software
1.4 Types of Data in Statistics and Machine Learning
1.5 Types of Machine Learning
1.6 Summary
1.7 Terminal Questions
1.8 Solutions/Answers

ignou
THE PEOPLE'S
UNIVERSITY

1.1 INTRODUCTION

Recall your childhood, how you learnt to recognise things. For example, your parents or some other person told you the name of the things around you dog or a cat (say), and you tried to identify features or patterns of the dog and cat so that next time you can name the dog or cat correctly. You know that this learning improves with experience.

Like this machine can also learn through data (for a machine, experience means data, the more data, the more learning). In this course, we will discuss some machine learning algorithms. You were expecting that this journey should start from the definition of machine learning. Keeping it in view, the definition and a very brief overview of the history of machine learning is discussed in Sec. 1.2. After going through the definition of machine learning in Sec. 1.2, the next thing that you should know in this journey is types of machine learning. But types of machine learning are directly related to some terminology used in measurement scales and types of data. So, first we have discussed the measurement scale and types of data in the Sections. 1.3 and 1.4, respectively. Finally, the types of machine learning is discussed in Sec. 1.5.

What we have discussed in this unit is summarised in Sec. 1.6. Self-Assessment Questions (SAQs) have been given in some sections, which are generally based on the content discussed in that section. But to give you a good practice of what we have discussed in this unit, some more questions based on the entire unit are given in Sec. 1.7 under the heading Terminal Questions. One of the characteristics of IGNOU study material is that it is self-contained. To make it self-contained regarding solutions of all the SAQ and Terminal Questions, we have given solutions of all such questions in Sec. 1.8. To best utilise this section, it is recommended that after going through all concepts discussed in this unit, first, you should try solutions of all SAQs and Terminal Questions yourself. In the case, your answer does not match the answer given in the solution of the same you may give it another try. If still not matching, you may refer to the given solution and identify the concept(s) where you need to improve.

In the next unit, you will study the evaluation of machine learning models.

Expected Learning Outcomes

After completing this unit, you should be able to:

- explain what is machine learning;
- explain different measurement scales and their corresponding names in the field of machine learning, data science and R software;
- explain types of data in the worlds of statistics and machine learning; and
- define types of machine learning.

1.2 DEFINITION OF MACHINE LEARNING AND ITS BRIEF HISTORICAL OVERVIEW

The term "machine learning" was coined by computer scientist Arthur Samuel in 1959. He defined machine learning as follows:

“The field of study that gives computers the ability to learn without being explicitly programmed”. … (1.1a)

The definition of machine learning by Tom M. Mitchell is given as follows:

“A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E”. … (1.1b)

Tom M. Mitchell gives the example of playing checkers to explain E, T and P as follows.

“**A checkers learning problem**:

- Task T: playing checkers
- Performance measure P: percent of games won against opponents
- Training experience E: playing practice games against itself”

After going through two definitions (1.1a) and (1.1b), you have understood that in machine learning, we give a task to a computer and it does not complete that work, but also makes decisions based on the patterns in data. This ability of taking correct decisions improves with experience. Here, by experience, we mean makes optimum decisions and do the assigned job.

A **brief historical overview of machine learning** is given as follows:

- In 1952, the world's first self-learning program to play games was created by Arthur Samuel, called the Samuel Checkers Playing Program.
- In 1958, inspiring from the function of the human mind, an early Artificial Neural Network (ANN), called perceptron was developed by Frank Rosenblatt that could learn from data.
- As mentioned in the definition of machine learning, the term "machine learning" was coined by computer scientist Arthur Samuel in 1959.
- In 1994, self-driving car road test was conducted of the double robot vehicles called Vita-2 and VaMP of Daimler-Benz and Ernst Dickman's.
- In 1997, Deep Blue (a supercomputer by IBM) beat Garry Kasparov, chess world champion of that time. It shows that machines can also be used to do that kinds of works where intelligence is required.
- Some of the measure developments during 2000 to 2017 are: Random Forest, AdaBoost, Naïve Bayes, k-means clustering, Deep learning, Natural language processing, Reinforcement learning, Dimensional reduction, etc.
- In 2018, Generative Pre-trained Transformer (GPT) was released.
- In 2019, GPT-2 was released.
- In 2020, GPT-3 was released.
- In 2022, GPT-3.5 was released.
- In 2023, GPT-4 (Chat GPT Plus) was released.
- In 2024, GPT-4 (GPT-4-turbo) was released.
- In 2025, GPT-5 was released.

Machine learning applications in various fields are increasing day by day. In short, we can say that a machine (computer) not only processes data faster and more accurately than humans, but in many cases, it has started to outperform humans in detecting objects, identifying patterns and solving many assigned tasks more effectively.

Now, you can try the following Self-Assessment Question.

SAQ 1

State whether the following statement is TRUE or FALSE. Give a reason in support of your answer.

GPT-3 was released in 2027.

1.3 MEASUREMENT SCALES IN STATISTICS, MACHINE LEARNING AND R SOFTWARE

Consider a data set given in Table 1.1 of 10 customers who visited a restaurant.

Table 1.1: Bill amount, tip amount and sex of 10 customers

Customer number	1	2	3	4	5	6	7	8	9	10
Bill amount	500	300	400	600	700	600	800	600	500	300
Tip amount	10	5	15	20	30	25	40	20	20	10
Sex	male	female	Female	male	male	male	male	female	male	male

From common understanding, you know that the tip amount generally depends on the bill amount. When such a relation exists between two variables, then you know that, in mathematics and statistics terminology, we call the bill amount an **independent variable** and the tip amount the **dependent variable**. You may refer to Remark 2 in Unit 1 of the course MST-011. … (1.2)

In machine learning language:

- The independent variable is known as a **feature** or **predictor,** or **covariate**. If we have more than one independent variable, then we will obviously call them features, predictors, or **covariates**. … (1.3)
- The dependent variable is known as the **target variable** or **outcome**. … (1.4)

So, in machine learning, we have one or more than one feature/features and also **may have** target variable, and our goal is to build an algorithm to predict the outcome or classification or making clusters for given values of features. But in machine learning, there are a large number of algorithms. So, a natural question that arises is how we decide which algorithm is appropriate. To understand the answer to this question, first, you should have an understanding of the measurement scale. So, in this section, first we will discuss measurement scales. There are four measurement scales which are generally used in statistics for counting or measuring some variable of interest. For example, if you want to know the number of pages in a notebook, you can easily count them. Also, if you want to know the weight of a person, you can easily measure it. But, in Statistics, act of counting and measurement is divided into 4 levels of measurement scales known as:

(1) Nominal Scale

(2) Ordinal Scale

(3) Interval Scale

(4) Ratio Scale

Let us discuss these scales of measurement one by one:

(1) Nominal Scale

In Latin, ‘Nomen’ means name. In nominal scale, we divide the objects under study into two or more categories by giving them unique names. The classification of objects into at least two or more categories is done in such a way that

(a) Each object takes place only in one category, i.e., each object falls in a unique category, i.e., it either belongs to a category or not. Mathematically, we may use the symbol (“=”, “$\neq$”) if an object falls in a category (=) or not ($\neq$).

(b) Number of categories must be sufficient to include all objects, i.e., there should not be scope for missing even a single object which does not fall in any of the categories. That is, in statistical language, categories must be mutually exclusive and exhaustive. … (1.5)

Generally, a nominal scale is used when we want to categorise the data based on a characteristic such as gender, race, region, religion, disease status (yes or no), etc. … (1.6)

To get more familiar with the idea of a nominal scale, let us consider some examples:

(i) Classification into Different Categories Based on Gender

This can be done by dividing the population into two categories male ‘M’ and female ‘F’

Category	**Name/Code**
Male	M
Female	F

Here we have named male as ‘M’ and female as ‘F’. This is not the only way, we can also code male by ‘0’ and female by ‘1’ or we may use any other convenient symbols. So, we note that the main thing is that we have to give a unique name to each category. Here, you cannot say that 1 > 0 because 0 and 1 do not represent numbers but they are behaving like names. … (1.7)

(ii) Classification into Different Categories Based on Caste

Different categories	Code allotted/Name given
General	Gen
Scheduled caste	SC
Scheduled tribes	ST
Backward class	BC
Others	‘O’

Here also, we can give a code to general, scheduled caste, scheduled tribes, backward class and other categories by ‘0’, ‘1’, ‘2’, ‘3’, ‘4’ respectively. Here also keep the same point in mind, we cannot say that $0 < 1$, 1 < 2, 2 < 3, 3 < 4 because they are playing the role of names, not numbers. … (1.8)

(iii) Classification into Different Categories Based on Number Allotted

In sports events, the numbers allotted to the participants also come under the nominal scale.

Note 1: We note that in the nominal scale, we have just coded the objects. Signs of less than (<) or greater than (>) do not make any sense in a nominal scale.

That is, we cannot talk about the order between two categories in the case of a nominal scale.

If in a measurement scale, order among different categories also makes sense, then this scale comes under the heading of an ordinal scale discussed as follows.

(2) Ordinal Scale

We have seen that order does not make any sense in nominal scale data. If order among different categories also makes sense, then the variable under study is considered an ordinal scale variable. So, in this scale < and > signs also make sense other than ("=", "$\neq$"). But here, we cannot find the actual difference between the two categories. … (1.9)

Generally, an ordinal scale is used when we want to measure the attitude scores towards the level of liking, satisfaction, preference, etc. Different designations in an institute can also be measured by using an ordinal scale. Stages of a disease, such as cancer (Stage I, Stage II, Stage III), also fall in the category of a nominal scale. … (1.10)

To get more familiar with the concept of an ordinal scale, let us consider some examples:

(i) Opinion of persons about the proposal of introducing co-education in a college comes under this scale. Suppose we assign '1'to strongly disagree, '2' to disagree '3' to indifferent (or neutral) '4' to agree and '5' to strongly agree. Here, the order also matters, and mathematically we may use the symbols >, < in addition to those used for nominal scale, i.e., =, $\neq$ as here strongly agree opinion comes first in order as compared to agree and so on, i.e., 5 > 4 > 3 > 2 > 1 or 1 < 2 < 3 < 4 < 5. But actual difference between different categories in this Likert Scale is not possible. This is because, suppose "strongly agree" means he/she gives marks from 75% to 100% for the co-education to be introduced and suppose "agree" means the marks are given in the range say 50% to 75%. Now, the actual difference between "strongly agree" and "agree" is not feasible in this sense.

$$\left[\begin{array}{l}\because \text{ "strongly agree" may have the marks percentage} \\ \text{as 80\% and "agree" marks 74\%. Similarly in other} \\ \text{case these values may be 90\% and 70\% respectively.}\end{array}\right]$$

(ii) In sixth pay commission, teachers of colleges and universities are designated as Assistant Professor, Associate Professor and Professor. The rank of Professor is higher than that of Associate Professor and designation of Associate Professor is higher than Assistant Professor. But you cannot find the actual difference between Professor and Associate Professor or Professor and Assistant Professor or Associate Professor and Assistant Professor. This is because, one teacher in a designation might have served certain number of years and have done a good quality of research work, etc. and other teacher in the same designation might have served for lesser number of years and have done unsatisfactory research work, etc. So, the actual difference between one designation and other designation cannot be found. So, one may be very near to his next higher designation and other may be very far from that person depending on their teaching experience and quality of teaching/research.

(iii) Based on the economic condition of a family, generally, families of a society are divided into three categories:

Higher-class family

Middle-class family

Lower-class family

Everybody knows that the economic condition of a higher-class family is better than middle-class family and a middle-class family is in a better condition compared to a lower-class family. But the actual difference between the economic condition of a higher-class family and a middle-class family or between a middle-class family and a lower-class family cannot be measured.

That is, we can only give order/rank to the three classes of the families but the actual difference cannot be measured. In all the above examples, the actual difference is not possible because all the ranks are on the ranges and not on fixed points.

Now, we are going to study the next higher level of measurement, wherein the actual differences can be found. This scale is known as an interval scale.

(3) Interval Scale

Nominal scale gives only names to the different categories, ordinal scale moves one step further, also provides the order between the categories and interval scale moves one step ahead and also provides the characteristic of the difference between any two categories. However, in an interval scale, 0 does not represent the absence of the quantity we are measuring. … (1.11)

An interval scale is used when we want to measure years/historical time/calendar time, temperature (except in the Kelvin scale), sea level, marks in the tests where there is negative marking, etc. Mathematically, this scale includes +, – in addition to >, < and $=, \neq$. … (1.12)

To get more familiar with the concept of interval scale, let us consider some examples:

(i) The measurement of time of a historical event comes under an interval scale because there is no fixed origin of time (i.e., ‘0’ year). As ’0’ years differ from calendar to calendar or society/country to society/country, e.g., Hindu, Muslim and Hebrew calendars have different origins of time, i.e., ‘0’ year is not well defined.

(ii) Measurement of temperature in degrees Celsius ($^{0}\mathrm{C}$) assumes $0^{0}\mathrm{C}$ when water starts freezing to ice, and it becomes ice at $-40^{\circ}\mathrm{C}$. So, in degrees Celsius, the origin is arbitrary, that is, why measurement of temperature in degrees Celsius comes in an interval scale. Because in the degree Celsius origin is arbitrary, we cannot say that $30^{0}\mathrm{C}$ is twice as hot as $15^{0}\mathrm{C}$. Because if it is so, then can we say that $4^{\circ}\mathrm{C}$ is -1 times $-4^{\circ}\mathrm{C}$? No, it is meaningless. Similarly, measurement of temperature in Fahrenheit comes in the interval scale.

(iii) Mean sea level (MSL) also has an arbitrary origin because it is the mean of two means, mean high tide and mean low tide (and mean high tide and mean low tide vary according to high and low pressure zones). Further, it also varies from place to place and time to time. So, measurement of sea level also comes in the interval scale.

(4) Ratio Scale

Ratio scale is the highest level of measurement because nominal scale gives only names to the different categories, ordinal scale provides orders between categories other than names, interval scale provides the characteristic of difference between categories other than names and orders, but the ratio scale has a natural zero (absolute zero) other than names, orders, and characteristic of difference between any two categories. In ratio scale, 0 represent the absence of the quantity we are measuring. … (1.13)

Ratio scale is used when we want to measure temperature in Kelvin, weight, height, length, age, mass, time, plane angle, etc. Ratio scale includes $\times$, $\div$ in addition to +, –, >, <, =, $\neq$. But be careful never take ‘0’ in the denominator while finding ratios. For example, $\frac{4}{0}$ is meaningless. … (1.14)

To get more familiar with the concept of ratio scale, let us consider some examples, where ratio scale is used:

(i) Measurement of temperature in the Kelvin scale comes under ratio scale because it has an absolute zero, which is equivalent to $-273.15^{0}\mathrm{C}$. This characteristic of origin allows us to make the statement like 50K (‘50K’ read as 50-degree Kelvin) is 5-time hot compared to 10K.

(ii) Counting of money also comes under ratio scale because it satisfies all the requirements of interval scale and has a natural zero. For example, suppose there are 60 teachers in a particular school in Delhi. If we associate a unique number to each teacher related to the cash, he/she has with him/her at the time of investigation. Then we have a fixed number corresponding to each teacher. Of course, two or more teachers may have the same cash. These teachers will be allotted the same number and will fall in one category. Here, we note that the numbers allotted to the teachers can be ordered, have an actual difference and also have an origin (i.e., absolute zero '0'). **Here natural zero indicates the absence of money in the pocket of the teacher**. If a teacher has Rs 500 and another teacher has Rs 100, then we can say that the teacher having Rs 500 has 5 times amount than a teacher having Rs 100. Thus, it satisfies all the requirements of ratio scale.

Remark 1: In some fields like epidemiology, biostatistics, etc., instead of interval and ratio scales, they measure the outcome as **discrete** and **continuous**. For more details, you may refer to 15th line onward from the last on page number 34 of Unit 2 of the course MST-019.

Permissible Statistical Tools

One of the advantages of measurement scales is that they help us to decide which statistical tool should be used in a given situation.

Table 1.2 shows the list of permissible statistical tools in the case of nominal, ordinal, interval and ratio scales. Based on information provided by these scales, their levels from lowest to highest are nominal, ordinal, interval and ratio (see Fig 1.1). That is why all the Statistical tools applicable on the lower scale will automatically be applicable on the next level scale. So, we will not repeat the permissible statistical tools used in the lower level scale. It is understood that statistical tools which are permissible for nominal will be permissible in case of ordinal scale and so on.

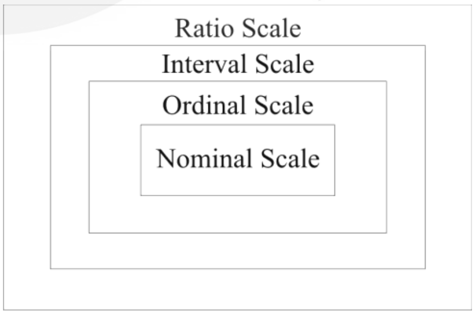


Fig. 1.1: Visualisation of the hierarchy of four measurement scales

Table 1.2

MEASUREMENT SCALE	PERMISSIBLE STATISTICAL TOOLS	LOGIC/REASON
NOMINAL SCALE	All statistical tools where only count (frequency) is required such as Mode, chi-square test, etc.	Here, counting is the only permissible operation.

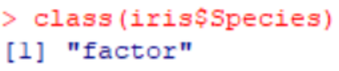
ORDINAL SCALE	All statistical tools where count (frequency) or order/rank or both are required such as Median all positional averages like quartile, Decile, percentile, Spearman's Rank correlation, etc.	Here, other than counting, order relation (less than or greater than) also exists.
INTERVAL SCALE	All statistical tools where count (frequency) or order/rank or difference between observations or any combination of these three are required, such as Mean, S.D., t-test, ANOVA, sample multiple and moment correlations, regression, etc.	Here, counting, order and difference operations hold.
RATIO SCALE	All statistical tools where count (frequency) or order/rank or difference between observations or product or quotient or any combination of these operations are required, such as Geometric mean (G.M.), Harmonic mean (H.M.), Coefficient of variation, etc.	Here, counting, order, difference, and natural zero exist.

Terminology used in Machine Learning and R Software Regarding Measurement Scale

The term "categorical variable" is used frequently in machine learning and data science. All variables measured using either a nominal or an ordinal scale of measurement are referred to as **categorical variables**. If we ask R software about the class of a categorical variable, it will tell us either **character** or **factor** or **ordered factor**. … (1.15)

For example,

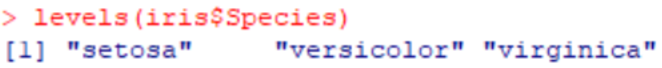
- Variable Species in the iris dataset in R is a categorical variable. Let us ask R about the class of this variable. Screenshot of R code and its output is shown as follows:



```
> class(iris$Species)
[1] "factor"
```

… (1.16)

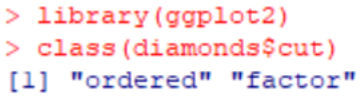
  If we also want to know the levels (names of different categories of the Species variable), then it can be obtained by running the following R code. The screenshot of R code and its output is shown as follows:



```
> levels(iris$Species)
[1] "setosa"     "versicolor" "virginica"
```

… (1.17)

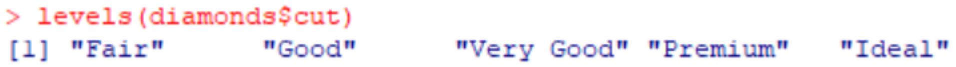
- The variable cut in the diamonds dataset in the ggplot2 package of R software is a categorical variable. Let us ask R about the class of this variable. Make sure that you have installed and loaded the ggplot2 package before running that code. In my system, the ggplot2 package is already installed, so I have to just load it using the library(ggplot2) command. Finally, a screenshot of R codes and their outputs is shown as follows:



```
> library(ggplot2)
> class(diamonds$cut)
[1] "ordered" "factor"
```

… (1.18)

If we also want to know the levels (names of different categories of the cut variable), then it can be obtained by running the following R code. The screenshot of R code and its output is shown as follows:



```
> levels(diamonds$cut)
[1] "Fair"      "Good"      "Very Good" "Premium"   "Ideal"
```

… (1.19)

Remark:

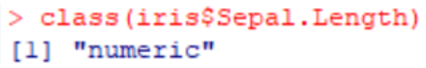
(1) There is no natural order (< or >) that makes sense in the three categories setosa, versicolor and virginica of the variable Species in the iris dataset in R. But by default, R software ordered the character vector in alphabetical order. So, in (1.17) setosa is written first, versicolor is written at the second position, and virginica is placed at the third position, which is nothing but simply alphabetical order. Here, order among the three categories setosa, versicolor and virginica does not make sense and so comes in a nominal scale of measurement. So, Species is a categorical variable. … (1.20)

(2) However, the five levels Fair, Good, Very Good, Premium and Ideal of the cut variable shown in (1.19) are ordered based on the grade of the cut from lowest to highest quality. Here, order makes sense and so comes in the ordinal scale of measurement. So, cut is also a categorical variable. … (1.21)

Out of four levels of measurement scales: nominal, ordinal, interval and ratio that are used in statistics, two (nominal and ordinal) of them have come under the umbrella of categorical variable. In machine learning and data science, the variables that fall into the other two scales (interval and ratio) are categorised as discrete and continuous. The good news is that we have already discussed discrete and continuous variables in Unit 2 of the course MST-011. So, we will not discuss all those details here again. It is recommended that if you are not familiar, you should go through the respective portion of Unit 2 of the course MST-011. If we ask R software about the class of a continuous variable, it will tell us **numeric**. If we ask R software about the class of a discrete variable, it will tell us either **numeric** or **integer**. … (1.22)

For example,

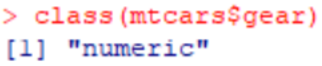
- Variable Sepal.Length in the iris dataset in R is a continuous variable. Let us ask R about the class of this variable. Screenshot of R code and its output is shown as follows:



```
> class(iris$Sepal.Length)
[1] "numeric"
```

… (1.23)

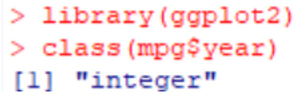
- Variable gear in the mtcars dataset in the base package of R software is a discrete variable which represents the number of gears in the cars. Let us ask R about the class of this variable. Screenshot of R code and its output is shown as follows:



```
> class(mtcars$gear)
[1] "numeric"
```

… (1.24)

- The variable year in the mpg dataset in the ggplot2 package of R software is a discrete variable. This variable represents the year of manufacture of car. Let us ask R about the class of this variable. Make sure that you have installed and loaded the ggplot2 package before running that code. In my system, the ggplot2 package is already installed, so I have to just load it using the library(ggplot2) command. Finally, a screenshot of R codes and their outputs is shown as follows:



```
> library(ggplot2)
> class(mpg$year)
[1] "integer"
```

… (1.25)

Now, you can try the following Self-Assessment Questions.

SAQ 2

In R, we have the mpg dataset in the ggplot2 package. One of the 11 columns of this dataset is cyl, which represents the number of cylinders in the car. Write the class of the cyl variable according to machine learning terminology.

SAQ 3

State whether the following statement is TRUE or FALSE. Give a reason in support of your answer.

The wt variable in mtcars dataset in R is a discrete variable.

SAQ 4

State whether the following statement is TRUE or FALSE. Give a reason in support of your answer.

We can obtain median of the Species variable in iris dataset in R.

1.4 TYPES OF DATA IN STATISTICS AND MACHINE LEARNING

To apply statistics tools or machine learning algorithms, we need data. So, let us first define data in statistics.

"The values of different objects collected in a survey or recorded values of an experiment over a time period taken together constitute what we call **data in Statistics**" … (1.26)

For example, the number of students present in 10 lectures one lecture per day on consecutive 10 working days out of a total strength of 40 students is given as follows:

40, 38, 35, 39, 37, 38, 39, 40, 33, 36

Each value in the data is known as **observation**. For example, in this data set 33 is an observation which tells us the information about the number of students present in the 9th day lecture. … (1.27)

Data based on the characteristic, nature of the characteristic, level of measurement, time and ways of obtaining/collecting it may be classified as follows:

- Quantitative data
- Qualitative data

} based on the characteristic

- Discrete data
- Continuous data

} based on nature of the characteristic

- Nominal data
- Ordinal data
- Interval data
- Ratio data

} based on level of measurement … (1.28)

- Time Series data
- Cross Sectional data

} based on time component

- Primary data
- Secondary data

} based on the ways of collecting the data

Let us briefly explain these different types of data one at a time:

Quantitative Data

Data are said to be quantitative if a numerical quantity (which exactly measures the characteristic under study) is associated with each observation. … (1.29)

Generally, interval or ratio scales are used as a measurement of scale in case of quantitative data. If we talk about machine learning or data science point of view then quantitative data may discrete or continuous. Data based on the following characteristics generally gives quantitative type of data: weight, height, age, length, area, volume, money, temperature, humidity, etc.

Qualitative Data

The quality of an object/thing cannot be measured numerically in exact terms. Thus, if the characteristic/attribute under study is such that it is measured only on the basis of presence or absence then the data thus obtained is known as qualitative data. … (1.30)

Generally, nominal and ordinal scales are used as a measurement of scale in case of qualitative data. If we talk about machine learning or data science point of view then qualitative data means it is a categorical variable. Data based on the following characteristics generally gives qualitative data: gender, marital status, qualification, colour, religion, satisfaction, types of trees, beauty, honesty, etc.

Discrete Data

If the nature of the characteristic under study is such that values of observations may be at most countable between two certain limits, then corresponding data are known as discrete data (concept of countability has already been discussed in Sec 2.3 of Unit 2 of the course MST-011). … (1.31)

For example,

- The number of books on the shelf of an Elmira in a library forms discrete data. Because the number of books may be 0 or 1 or 2 or 3,….
- If there are 30 students in a class, then the number of students present in a lecture forms discrete data. Because the number of present students may be 1 or 2 or 3 or 4 or…or 30.
- Number of mistakes on a particular page of a book. Obviously, the number of mistakes may be 0 or 1 or 2 or 3….

Continuous Data

Data under study are said to be continuous if the measurement of the observations of a characteristic under study may be any real value between two certain limits. … (1.32)

For example,

- Data obtained by measuring the heights of the students of a class of say 30 students form continuous data, because if the minimum and maximum heights are 152cm and 175 cm, then the heights of the students may take any possible values between 152 cm and 175 cm. For example, it may be 152.2375 cm, 160.31326… cm, etc.
- Data obtained by measuring the weights of the students of a class also form continuous data because the weights of students may be 48.25796…kg, 50.275kg, 42.314314314…kg, etc.

Nominal Data

Data collected using a nominal scale is called nominal data.

Similarly, data collected using the ordinal scale, interval scale and ratio scale are called **ordinal data, interval data** and **ratio** scale **data,** respectively. These scales of measurement have already been discussed in detail in the previous section. … (1.33)

Time Series Data

Data are said to be time series data if time is one of the main variables and the data is collected usually at regular intervals of time related to the characteristic(s) under study, and show how characteristic(s) change over time. … (1.34)

For example, quarterly profit of a company for the last eight quarters, yearly production of a crop in India for the last six years, yearly expenditure of a family on different items for the last five years, weekly rate of inflation for the last ten weeks, etc., all form time series data.

Note 2: If the purpose of the data collection has its connection with geographical location, then it is known as **Spatial Data.**

For example,

- Price of petrol in Delhi, Haryana, Punjab, and Chandigarh at a particular time.
- Number of runs scored by a batsman in different matches in a one-day series in different stadiums.

Note 3: If the purpose of the data collection has its connection with both time and spatial (space or location), then it is known as **Spatio-Temporal Data.**

For example, data related to the population of different states in India in 2001 and 2011 is an example of spatial-temporal data.

Note 4: In time series data, spatial data and spatio-temporal data, we see that the concept of frequency has no significance and hence known as **non-frequency data**. For instance, in the example discussed in the case of time series data, expenditure of Rs 40000 on food in 2006 is itself important; here, its frequency, say 3 (repeated three times), does not make any sense.

Note 5: Now consider the case of marks of 40 students in a class out of 10 (say). Here we note that there may be more than one student who scores the same marks in the test. Suppose out of 40 students, 5 score 10 out of 10, which means the mark 10 has frequency 5. This type of data, where frequency is meaningful, is known as **frequency data.**

Cross-sectional Data

Sometimes we are interested in knowing how a characteristic (such as income or expenditure, population, votes in an election, etc.) under study at one point in time is distributed over different subjects (such as families, countries, political parties, etc.). This type of data, which is collected at one point in time, is known as cross-sectional data. … (1.35)

For example, the annual income of different families of a locality, a survey of consumers' expenditure conducted by a research scholar, opinion polls conducted by an agency, salaries of all employees of an institute, etc. are examples of cross-sectional data.

Note 6:

- If you are interested in knowing the changes in a characteristic, say the expenditure of a family over a period of time, then you have to use time series data.
- If you are interested in knowing the changes in a characteristic, say expenditure of different families at a single point in time, you have to use cross-sectional data.

Primary Data

Data which are collected by an investigator or agency or institution for a specific purpose are known as primary data for these people. … (1.36)

For example, if a hospital records the data of the patients visited to that hospital regarding their sex, age, disease, treatment given, etc., then for that hospital, this collected data will be considered as primary data.

Secondary Data

Data obtained/gathered by an investigator or agency, or institution from a source which already exists are called secondary data. … (1.37)

For example, if a research scholar collects data from the website of an institute for his/her research, then the data he/she is using will be called secondary data.

Data in Machine Learning

Understanding data types in machine learning is essential to choosing and building an appropriate machine learning algorithm. In machine learning, data have a broad meaning. You can understand it through the following examples of data in machine learning. … (1.38)

Images of objects (such as images of handwritten digits), faces, pictures, textual data, social media posts, videos, audios, customers' reviews, product ratings, emails, GPS data, sensor data, patient’s records, fruit types, etc.(1.39)

Now, you can try the following Self-Assessment Question.

SAQ 5

In R we have a built-in dataset iris. One of the five columns of the dataset is Species having 50 entries for each of the three categories: setosa, versicolor and virginica. Species column falls in which level of measurement scale.

1.5 TYPES OF MACHINE LEARNING

Broadly, machine learning is classified into two categories, known as supervised and unsupervised machine learning. However, there are two more categories of machine learning other than supervised and unsupervised, known as semi-supervised and reinforcement learning. In this section, we will discuss them with examples. So, let us discuss them one at a time as follows:

Supervised Machine Learning

In machine learning, algorithms are built with data. So, first we should have a look on data. In machine learning data in general look like this:

Table 1.3: Typical dataset with p features and target (outcome) variable Y

Target variable Or Outcome variable Y	**Feature variable** X_1	**Feature variable** X_2	**Feature variable** X_3		**Feature variable** X_p
y_1	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	…	$x_{1,p}$
y_2	$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	…	$x_{2,p}$
y_3	$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	…	$x_{3,p}$
⋮	⋮	⋮	⋮	⋮	⋮
y_n	$x_{n,1}$	$x_{n,2}$	$x_{n,3}$	…	$x_{n,p}$

So, in Table 1.3:

1st column represents the target variable, and it is generally denoted by Y.

2nd column represents 1st feature variable, and it is generally denoted by X_1.

3rd column represents 2nd feature variable, and it is generally denoted by X_2.

4th column represents 3rd feature variable, and it is generally denoted by X_3.

and so on

$(p+1)^{th}$ column represents p^{th} feature variable, and it is generally denoted by X_p.

Each row of Table 1.3 is known as an instance.

Labelled Dataset: If a dataset also contains a target variable with features, then it is known as a labelled dataset. For example, data in Table 1.3 is a labelled dataset. … (1.40)

Unlabeled Dataset: If a dataset does not contain a target variable with features, then it is known as an unlabeled dataset. For example, if we remove the first column of Table 1.3, then the reduced dataset will be known as an unlabeled dataset. … (1.41)

ignou
THE PEOPLE'S
UNIVERSITY

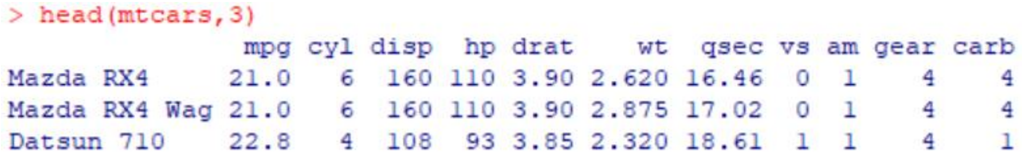
Now, we can define supervised machine learning as follows.

If a machine learning algorithm is trained using a labelled dataset, then it is known as a **supervised machine learning algorithm**. All such machine learning algorithms which are trained using a labelled dataset together constitute a type of machine learning known as **supervised machine learning**. So, supervised machine learning is a type of machine learning where algorithms are trained on a labelled dataset. … (1.42)

On the basis of the target variable, supervised machine learning is further classified into two categories, known as regression and classification, defined as follows:

Regression: If the target variable is continuous, then the machine learning algorithm is known as a regression problem in supervised machine learning. … (1.43)

For example, in R, we have a built-in dataset, mtcars. The first three rows of this dataset can be seen by running the following code on R console. Screenshot with output is shown as follows:



```
> head(mtcars,3)
              mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4     21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag 21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710    22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
```

… (1.44)

If we consider the variable:

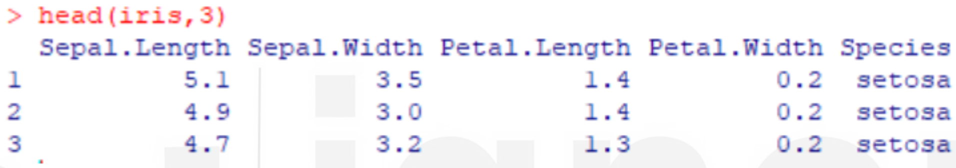
- mpg (mile per gallon) as the target variable, which represents the fuel efficiency of different cars and therefore is a continuous variable, and

- hp (Gross horsepower) as one feature,

then training an algorithm to predict mpg for given values of hp is a regression problem in supervised machine learning.

Classification: If the target variable is categorical, then the machine learning algorithm is known as a classification problem in supervised machine learning.

For example, in R, we have a built-in dataset, iris. The first three rows of this dataset can be seen by running the following code on R console. Screenshot with output is shown as follows:



```
> head(iris,3)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
```

… (1.45)

If we consider the variable:

- Species (which specifies the corresponding name of the iris flower: setosa or versicolor or virginica and hence is a categorical variable) as the target variable, and

- Sepal.Length, Sepal.Width, Petal.Length and Petal.Width as four features

then training an algorithm to predict the species of the flower for a given value of the four features is a classification problem in supervised machine learning.

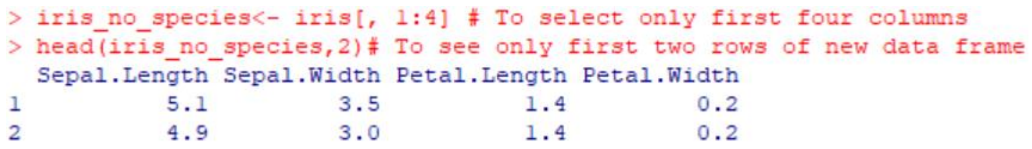
Now, we define unsupervised machine learning as follows.

Unsupervised Machine Learning

We have already discussed labelled and unlabeled datasets. So, we can easily define unsupervised machine learning.

If a machine learning algorithm is trained using an unlabeled dataset, then it is known as an **unsupervised machine learning algorithm**. All such machine learning algorithms which are trained using an unlabeled dataset together constitute a type of machine learning known as **unsupervised machine learning**. So, unsupervised machine learning is a type of machine learning where algorithms are trained on an unlabeled dataset. … (1.46)

For example, in R, we have a built-in dataset, iris. You have already seen the first three rows of this dataset (you may refer to 1.45). Suppose the column of Species is removed from this dataset. Now, we are left with only four features. This can be done as follows:



```
> iris_no_species<- iris[, 1:4] # To select only first four columns
> head(iris_no_species,2)# To see only first two rows of new data frame
  Sepal.Length Sepal.Width Petal.Length Petal.Width
1          5.1         3.5          1.4         0.2
2          4.9         3.0          1.4         0.2
```

… (1.47)

In this new data frame iris_no_species we do not have the target variable Species. Using this dataset, if we want to train an algorithm to classify the data into different categories, then such types of algorithms fall in the category of unsupervised machine learning.

We will do this job using K-Means Clustering in Sec. 6.3 in Unit 6 of this course.

Semi-Supervised Machine Learning

We have studied two extreme cases when we have:

- completely labelled data and … (1.48)
- totally unlabeled data. … (1.49)

Now, in semi-supervised machine learning, we need only partially labelled data.

If a machine learning algorithm is trained using a partially labelled dataset, then it is known as a **semi-supervised machine learning algorithm**. Here, by partially labelled data means most of the data is unlabeled, only a small fraction (may be 2% or 5% or 10%, etc.) of the data is labelled. So, semi-supervised machine learning is a type of machine learning where algorithms are trained on a partially labelled dataset. … (1.50)

Semi-supervised machine learning is beyond the scope of the course, so we will not discuss it further.

Reinforcement Machine Learning

Reinforcement learning is a kind of trial-and-error learning. If the **agent** (technical name for the decision maker that/who performs actions) interacts and explores the **environment** (technical name for the system in which the agent performs) and makes decisions, and in return agent gets a reward (technical name for reward or penalties) as per policies (technical name for rules to optimise rewards). … (1.51)

Reinforcement machine learning is beyond the scope of the course, so we will not discuss it further.

Let us do some examples on the types of machine learning.

Example 1: Out of 1000 rows, only the first 4 rows of a dataset are shown as follows:

Area (in Square yards)	**Number of rooms**	**Location**	**Price (in lakhs rupees)**
200	2	Urban	250
250	3	Urban	300
150	2	Urban	170
200	2	Suburban	225

Suppose you are interested in predicting the price of a house with having area of 180 square yards, 2 rooms and located in an urban location. Answer the following questions:

(a) In which types of machine learning does this problem belong to: supervised, unsupervised, semi-supervised or reinforcement learning problem. Explain.

(b) Name the features and the target variable.

(c) Further classify part (a).

Solution: (a) In this problem, we have both features and a target variable, so we have a labelled dataset. Hence, it is a supervised machine learning problem.

(b) In this problem, we have three features: (i) Area (ii) number of rooms (iii) location and price of the house is the target variable.

(c) In part (a), we have explained why it is a supervised machine learning problem. We know that if the target variable is continuous, then the underlying problem is considered a regression problem, and if the target variable is categorical, then it is considered a classification problem. Here, the target variable is the price of a house, which is a continuous variable, so it is a regression problem in supervised machine learning.

Now, you can try the following Self-Assessment Question.

SAQ 6

State whether the following statement is TRUE or FALSE. Give a reason in support of your answer.

A learning which involves an agent that learns by interacting with an environment, receiving rewards or penalties, is called semi-supervised machine learning.

1.6 SUMMARY

A brief summary of what we have covered in this unit is given as follows:

- The term "machine learning" was coined by computer scientist Arthur Samuel in 1959. He defined machine learning as follows:

"The field of study that gives computers the ability to learn without being explicitly programmed".

- The definition of machine learning by Tom M. Mitchell is given as follows:

  "A computer program is said to **learn** from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"

- In machine learning language:
  - The independent variable is known as a **feature** or **predictor,** or **covariate**. If we have more than one independent variable, then we will obviously call them features, predictors, or **covariates**.
  - The dependent variable is known as the **target variable** or **outcome**.
- **Nominal Scale**: In Latin, 'Nomen' means name. In nominal scale, we divide the objects under study into two or more categories by giving them unique names.
- **Ordinal Scale**: We have seen that order does not make any sense in nominal scale data. If order among different categories also makes sense, then the variable under study is considered an ordinal scale variable. So, in this scale < and > signs also make sense other than ("=", "$\neq$"). But here, we cannot find the actual difference between the two categories.
- **Interval Scale**: Nominal scale gives only names to the different categories, ordinal scale moves one step further, also provides the order between the categories and interval scale moves one step ahead and also provides the characteristic of the difference between any two categories. However, in an interval scale, 0 does not represent the absence of the quantity we are measuring.
- **Ratio Scale**: Ratio scale is the highest level of measurement because nominal scale gives only names to the different categories, ordinal scale provides orders between categories other than names, interval scale provides the characteristic of difference between categories other than names and orders, but the ratio scale has a natural zero (absolute zero) other than names, orders, and characteristic of difference between any two categories. In a ratio scale, 0 represents the absence of the quantity we are measuring.
- **Discrete and Continuous Outcomes**: In some fields, like epidemiology, biostatistics, etc., instead of interval and ratio scales, they measure the outcome as **discrete** and **continuous**. For more details, you may refer to 15^{th} line onward from the last on page number 34 of Unit 2 of the course MST-019.
- **Categorical Variable or Factor in R**: The term "categorical variable" is used frequently in machine learning and data science. All variables measured using either a nominal or an ordinal scale of measurement are

referred to as **categorical variables**. If we ask R software about the class of a categorical variable, it will tell us either **character** or **factor** or **ordered factor**.

- **Data**: The values of different objects collected in a survey or recorded values of an experiment over a time period taken together constitute what we call **data in Statistics.**

- **Quantitative Data**: Data are said to be quantitative if a numerical quantity (which exactly measures the characteristic under study) is associated with each observation.

- **Qualitative Data**: The quality of an object/thing cannot be measured numerically in exact terms. Thus, if the characteristic/attribute under study is such that it is measured only on the basis of presence or absence, then the data thus obtained is known as qualitative data.

- **Discrete Data**: If the nature of the characteristic under study is such that values of observations may be at most countable between two certain limits, then corresponding data are known as discrete data.

- **Continuous Data**: Data under study are said to be continuous if the measurement of the observations of a characteristic under study may be any real value between two certain limits.

- **Cross-sectional Data**: Sometimes we are interested in knowing how a characteristic (such as income or expenditure, population, votes in an election, etc.) under study at one point in time is distributed over different subjects (such as families, countries, political parties, etc.). This type of data, which is collected at one point in time, is known as cross-sectional data.

- **Primary Data**: Data which are collected by an investigator or agency or institution for a specific purpose are known as primary data for these people.

- **Secondary Data**: Data obtained/gathered by an investigator or agency, or institution from a source which already exists are called secondary data.

- **Data in Machine Learning**: In machine learning, data have a broad meaning. You can understand it through the following examples of data in machine learning.

  Images of objects (such as images of handwritten digits), faces, pictures, textual data, social media posts, videos, audios, customers' reviews, product ratings, emails, GPS data, sensor data, patient’s records, fruit types, etc.

- **Labelled Dataset**: If a dataset also contains a target variable with features, then it is known as a labelled dataset.

- **Unlabeled Dataset**: If a dataset does not contain a target variable with features, then it is known as an unlabeled dataset.
- **Supervised machine learning** is a type of machine learning where algorithms are trained on a labelled dataset.
- **Regression**: If the target variable is continuous, then the machine learning algorithm is known as a regression problem in supervised machine learning.
- **Classification**: If the target variable is categorical, then the machine learning algorithm is known as a classification problem in supervised machine learning.
- **Unsupervised machine learning** is a type of machine learning where algorithms are trained on an unlabeled dataset.
- **Semi-supervised machine learning** is a type of machine learning where algorithms are trained on a partially labelled dataset.
- **Reinforcement learning** is a kind of trial-and-error learning. If the **agent** (technical name for the decision maker that/who performs actions) interacts and explores the **environment** (technical name for the system in which the agent performs) and makes decisions, and in return agent gets reward (technical name for reward or penalties) as per policies (technical name for rules to optimise rewards).

1.7 TERMINAL QUESTIONS

1. State whether the following statement is TRUE or FALSE. Give a reason in support of your answer.

   drv variable in the mpg dataset of the ggplot2 package is an interval scale variable.

2. State whether the following statement is TRUE or FALSE. Give a reason in support of your answer.

   Machine learning algorithms can work only with numerical data.

3. State whether the following statement is TRUE or FALSE. Give a reason in support of your answer.

   Siri (speech recognition) is a continuous variable.

4. Suppose in a study, we have a gender variable where male and female are denoted by 0 and 1, respectively. Can we obtain the mean of the variable gender in this dataset?

5. State whether the following statement is TRUE or FALSE. Give a reason in support of your answer.

   Suppose you have 100000 images, out of which only 2000 are labelled dog or cat, then it is a problem of reinforcement learning.

6. Out of 10000 customer data, only the first 4 customers' data regarding their credit card annual spending are shown as follows:

Customer ID	Annual spending on credit cards (in rupees)	Number of transactions
1	1000000	25
2	600000	10
3	2500000	80
4	1200000	20

The bank wants to group cardholders as high spenders versus low spenders so that the bank can give them some offers accordingly. Answer the following questions.

(a) Name the type of machine learning this problem belongs to: supervised, unsupervised, semi-supervised or reinforcement learning problem. Explain.

(b) Name the feature variables.

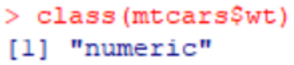
1.8 SOLUTIONS/ANSWERS

Self-Assessment Questions (SAQs)

1. It is a false statement because GPT-3 was released in 2020 not in 2017.

2. In R, the cyl variable of the dataset mpg in the ggplot2 package has 4 discrete values: 4, 5, 6 and 8. So, class of the cyl variable in the terminology of machine learning is discrete.

3. It is a false statement because the variable wt in mtcars dataset in R is a continuous variable instead of discrete because it can take any possible value between minimum and maximum weights. While in the case of discrete variable it can take at the most countable values. If we ask this question to R software then it should give answer as numeric. Screenshot of R code and its output is shown as follows.



```
> class(mtcars$wt)
[1] "numeric"
```

4. It is a false statement. In R, the Species column of the iris dataset has three categories: setosa, versicolor and virginica. Since setosa, versicolor and virginica are the names of three flowers so we cannot talk about the < or > operations on them. Hence, Species column falls in the nominal scale of measurement. But to obtain median, we should have a variable of at least ordinal scale. So, we cannot obtain the median of the variable Species in the iris dataset in R.

5. The Species column of the iris dataset has three categories: setosa, versicolor and virginica. Since setosa, versicolor, and virginica are the

names of three flowers so we cannot talk about the less than (<) or greater than (>) operations on them. Hence, the Species column falls in the nominal scale of measurement.

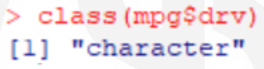
6. It is a false statement because it happens in reinforcement machine learning, so it is reinforcement machine learning instead of semi-supervised machine learning.

Terminal Questions

1. It is a false statement because the variable drv gives the information whether the engine of the car delivers power to the:

- front wheels (“f”) or
- rear-wheel (“r”) or
- all four wheels (“4”)

So, this variable is of nominal scale in the terminology of statistics, in the terminology of machine learning, it is a categorical variable, and in the terminology of R software it is a character vector. If we ask this question to R software then as mentioned earlier it should give answer as character. Screenshot of R code and its output is shown as follows.



```
> class(mpg$drv)
[1] "character"
```

2. It is a false statement because machine learning algorithms can handle non-numerical data also such as categorical, text, image, etc.

3. It is a false statement because Siri converts speech to text and identifies which words or phrases were spoken. So, it is a classification problem because the words that will be used in a speech are selected from a finite list of words of that language. So, the words and phrases used in a speech form different classes.

4. In this dataset, values 0 and 1 are names of two different categories of the gender variable, not the numbers. Therefore, we cannot obtain the mean of the variable 'gender'. To obtain the mean of a variable, it should be at least of an interval scale but here the gender variable is a nominal scale variable.

5. It is a false statement because we know that if we have:

- complete labelled dataset, then it is known as supervised machine learning.
- a totally unlabelled dataset, then it is known as unsupervised machine learning.
- partially labelled dataset, then it is known as semi-supervised machine learning.

Hence, the right answer is given situation is of semi-supervised machine learning.

6. (a) In this problem, the target variable is to classify customers into two groups: high spenders and low spenders. This variable is not present in the given dataset. Hence, it is an unsupervised machine learning problem.

   (b) In this unsupervised machine learning problem, we have only two feature variables: (i) annual spending and (ii) number of transactions made during the year.

UNIT 2

EVALUATION METRICS FOR MACHINE LEARNING MODELS

Structure

2.1 Introduction

Expected Learning Outcomes

2.2 Partition of a Dataset into Training and Testing Datasets

2.3 Overall Accuracy of an Algorithm

2.4 Confusion Matrix and Evaluation Metrics

2.5 Summary

2.6 Terminal Questions

2.7 Solutions/Answers

2.1 INTRODUCTION

In the previous unit, you studied types of machine learning. Like me, you were also eagerly waiting to see how a machine can learn from data. In Sec. 2.3, you get a tiny flavour of that. But the actual objective of this unit is to explain some performance measures of evaluating a machine learning algorithm on the basis of which we can say that one algorithm is better than the other. To do so, first, we have to build at least two machine learning algorithms so that we can compare their performance measures. In machine learning terminology, performance measures are known as evaluation metrics. However, before doing all this, first we have to define training and testing datasets and the same is done in Sec. 2.2. After that some evaluation metrics are discussed in Secs. 2.3 and 2.4.

What we have discussed in this unit is summarised in Sec. 2.5. Self-Assessment Questions (SAQs) have been given in some sections which are generally based on the content discussed in that section. But to give you a good practice of what we have discussed in this unit, some more problems based on the entire unit are given in Sec. 2.6 under the heading Terminal Questions. Due to the reason mentioned in Sec. 1.1 of Unit 1 of this course, solutions of all the SAQs and Terminal Questions are given in Sec. 2.7.

In the next unit, you will study the OC curve and the area under the curve (AUC) in detail.

Expected Learning Outcomes

After completing this unit, you should be able to:

- explain training and testing datasets;
- obtain a confusion matrix of an algorithm and some other evaluation metrics such as sensitivity, specificity, positive predictive value, negative predictive value, overall accuracy, balanced accuracy, etc.

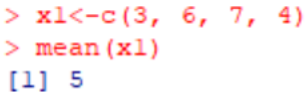
2.2 PARTITION OF A DATASET INTO TRAINING AND TESTING DATASETS

From Sec. 1.2 of the previous unit, recall that the term "machine learning" was coined by computer scientist Arthur Samuel in 1959, and he defined machine learning as follows:

“The field of study that gives computers the ability to learn without being explicitly programmed”. … (2.1)

If you know the meaning of “explicitly programmed” used in this definition, then good, but if you do not know its meaning, very good. Here, by very good, I mean do not lose heart, you are reading IGNOU study learning material (SLM) which is known for its three features: self-explanatory, self-contained and teacher-built-in. So, if you go through the SLM sequentially, then you will get the answers to most of your queries related to the learning part of the topics discussed in SLM. … (2.2)

Now, come to the point of what the meaning of **explicitly programmed** is. It means you have a complete flow chart for every possible situation and clear-cut instructions for each situation on what to do. The job of a computer is just to follow a given set of rules and perform operations accordingly. The list of all rules and operations in a logical sequence is prepared by a traditional computer programmer. For example, if you want to obtain the mean of the numbers 3, 6, 7, 4 using R software, then it can be done as follows:



```
> x1<-c(3, 6, 7, 4)
> mean(x1)
[1] 5
```

Now, here, first of all, the four numbers 3, 6, 7, 4 are stored in a vector x1. After that, we have applied the built-in function mean(). The programming of mean() function is explicitly programmed in the R software, i.e., the system will perform the operation $\frac{\text{Sum of all observations}}{\text{Number of observations}}$. So, hope this simple example has explained the meaning of explicitly programmed. … (2.3)

In this definition, the second important thing that you should know is the meaning of “**ability to learn**”. The learning part in machine learning algorithms is dictated by the dataset, and the more data we have, the more the machine learning algorithm learns. So, the algorithm’s learning ability increases with more and more data we have to train the algorithm. So, the moral of the story is we need a **good amount of data not only for training an algorithm**, but we also need a **good amount of data to test the trained algorithm**. … (2.4)

To define these two datasets (i) Training dataset and (ii) Testing dataset is the objective of this section. So, it is the right place to meet that objective. This classification of the original dataset into two datasets: training and testing datasets is visualised in Fig. 2.1.

Training Dataset: The original dataset is partitioned into two datasets generally in the ratio 80 : 20 or 75 : 25 or 70 : 30. That is, the larger dataset contains 70 to 80% of the original dataset. This larger dataset, which is a random subset of the original dataset, is used to train the machine learning algorithm and is known as the training dataset. A machine learning algorithm tries to find insights into the training dataset and discovers patterns in the training dataset. So, the more training data an algorithm has, the more learning will take place and therefore more accuracy will be there when we will apply this trained algorithm to an unseen dataset. … (2.5)

Testing Dataset: The original dataset is partitioned into two datasets generally in the ratio 80 : 20 or 75 : 25 or 70 : 30. That is, the smaller dataset contains 20 to 30% of the original dataset. This smaller dataset, which is a random subset of the original dataset, is used to test the trained machine learning algorithm and is known as the test dataset. The test dataset is used to determine the performance of the trained model on an unseen dataset. … (2.6)

Before closing this section, one important point that you should keep in mind is that the difference between the performance measures of an algorithm between the training dataset and testing dataset should be less. If the algorithm performs well in the training dataset but poorly in the testing dataset, it is an indication of the problem of **overfitting**. … (2.7)

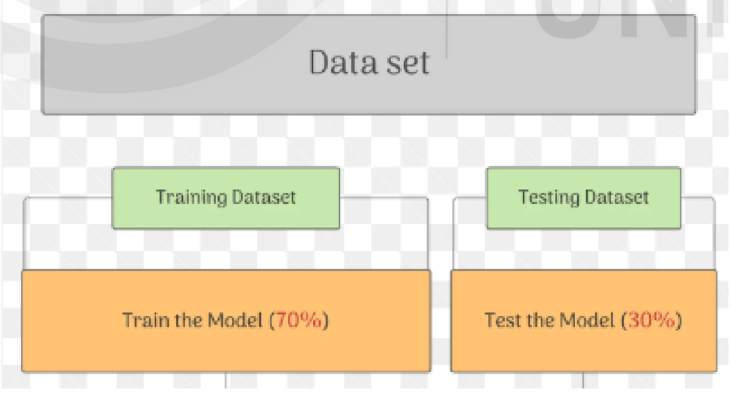


Fig. 2.1: Visualisation of the classification of the original dataset into two datasets

Now, you can try the following Self-Assessment Question.

SAQ 1

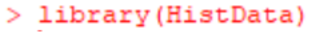
State whether the following statement is TRUE or FALSE. Give a reason in support of your answer.

Generally, a testing dataset is a random subset of the original dataset and contains 20 to 30% instances of the original dataset.

2.3 OVERALL ACCURACY OF AN ALGORITHM

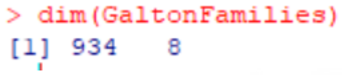
As mentioned in Sec. 2.1 of this unit, the main objective of this unit is to explain some evaluation metrics of machine learning algorithms. We want to do it so that later on, we can use these metrics to compare two machine learning algorithms to determine which is better than the other with respect to a metric. So, first, we have to build two machine learning algorithms.

Let us consider the GaltonFamilies dataset in the HistData package of R. I have already installed this package in my system. So, I have to just load this package in my system and the same can be done as follows.



```
> library(HistData)
```

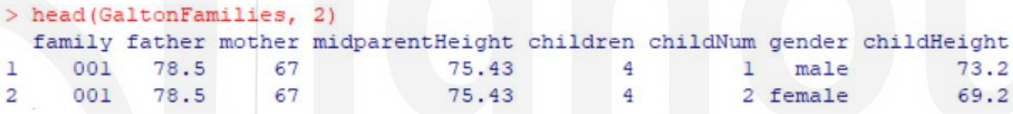
After that, we have to check how many rows and columns are in the GaltonFamilies dataset, which can be checked as follows.



```
> dim(GaltonFamilies)
[1] 934   8
```

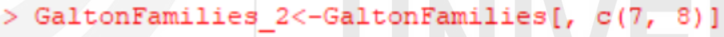
… (2.8)

We see that there are 934 rows and 8 columns in this dataset. To see the first two rows of this dataset, we can run the following R code.



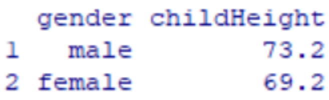
```
> head(GaltonFamilies, 2)
  family father mother midparentHeight children childNum gender childHeight
1    001   78.5     67           75.43        4        1   male        73.2
2    001   78.5     67           75.43        4        2 female        69.2
```

To give you a tiny flavour of a machine learning algorithm, we are going to build a machine learning algorithm where the sex of a child will be predicted on the basis of the given height of the child. So, we need only two variables, gender and childHeight, out of the 8 variables in this dataset. This can be done as follows.



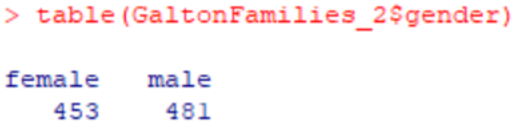
```
> GaltonFamilies_2<-GaltonFamilies[, c(7, 8)]
```

Now, let us see the first two rows of this new data frame, which has only two columns.



```
  gender childHeight
1   male        73.2
2 female        69.2
```

Great, we need only two variables in this new dataset. Now, let us see how many male and female children there are. This can be obtained by running the following R code.



```
> table(GaltonFamilies_2$gender)

female   male
   453    481
```

… (2.10)

From a machine learning algorithm point of view, it is great that the ratio of male and female children is almost equal. In other words, we can say that this dataset is not biased from a gender point of view.

But dear learner, I want to explain an issue that you may face in your career as a data scientist. To explain that point, I have to start with a biased dataset with respect to some variable. The technical word that is used to express this fact is prevalence, which you have already studied in Section 2.4 of Unit 2 of the course MST-019 (you may refer to page number 38). In this dataset prevalences of male and female children are given by … (2.11)

$$\text{Prevalence of male children in the study} = \frac{\text{Number of male children}}{\text{Total number of children}}$$

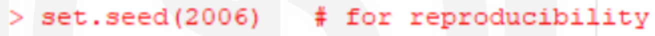
$$= \frac{481}{453+481} = \frac{481}{934} = 0.5149893$$

$$\text{Prevalence of female children in the study} = \frac{\text{Number of male children}}{\text{Total number of children}}$$

$$= \frac{453}{453+481} = \frac{453}{934} = 0.4850107$$

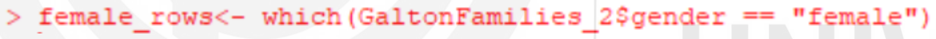
So, as mentioned earlier, the prevalence of male and female children is almost equal. This much difference is considered ok. But we want to make it biased with respect to the gender variable, so that as a data scientist, you should have an idea of the effect of prevalence in building a machine learning algorithm. So, to make it biased with respect to the gender variable, let us remove 300 rows of female children randomly from this dataset. This can be done as follows. … (2.12)

First of all, we have to set a seed for reproducibility purposes.



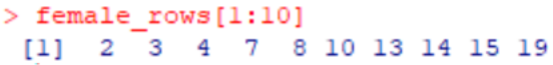
```
> set.seed(2006)    # for reproducibility
```

Next, we have to identify the positions of rows corresponding to female children. This can be done as follows.



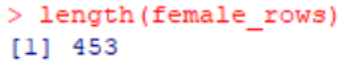
```
> female_rows<- which(GaltonFamilies_2$gender == "female")
```

So, positions of rows of female children have been stored in the variable female_rows. If we want to see the first 10 positions of the female children, it can be seen by running the following R code.



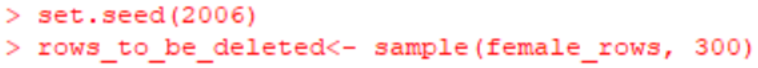
```
> female_rows[1:10]
 [1]  2  3  4  7  8 10 13 14 15 19
```

Since the dataset GaltonFamilies_2 has 453 female children (you may refer to 2.10), so we expect that there should be 453 elements in the vector female_rows. Let us verify it.



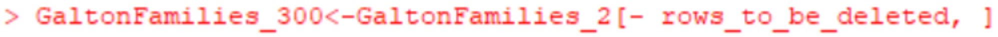
```
> length(female_rows)
[1] 453
```

So, verification is done. Next, we have to select 300 row positions randomly out of 453, and it can be done using the sample() function as follows.



```
> set.seed(2006)
> rows_to_be_deleted<- sample(female_rows, 300)
```

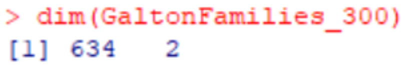
Next, we have to remove rows corresponding to these randomly selected 300 rows from the data frame GaltonFamilies_2, and it can be done as follows.



```
> GaltonFamilies_300<-GaltonFamilies_2[- rows_to_be_deleted, ]
```

ignou
THE PEOPLE'S
UNIVERSITY

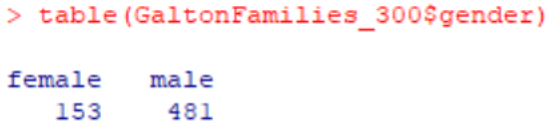
Now, we expect that the number of rows in GaltaonFamilies_300 should be 634 (= 934 – 300). Let us check it.



```
> dim(GaltonFamilies_300)
[1] 634   2
```

… (2.13)

Let us now check the number of male and female children in this dataset.

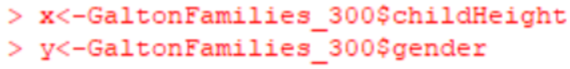


```
> table(GaltonFamilies_300$gender)

female   male
   153    481
```

… (2.14)

Finally, we now have a data frame GaltonFamilies_300, to work with.

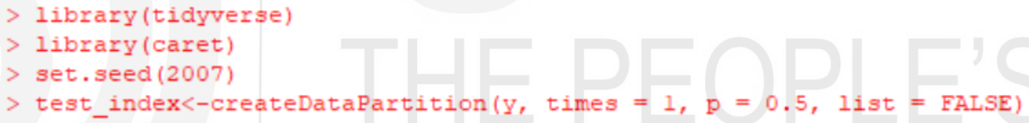
In usual notations, we denote the feature variable by x and the target variable by y. Let us store values in x and y as follows.



```
> x<-GaltonFamilies_300$childHeight
> y<-GaltonFamilies_300$gender
```

… (2.15)

Next, to split the dataset GaltonFamilies_300 into train and test datasets, we will use createDataPartition() function of the caret package. I have already installed the caret package, so I have to just load it. Also, we will use the pipe operator, which lies in the package tidyverse. I have already installed the tidyverse package. So, I have to just load it. This time, if we want, we can use a different seed before running the createDataPartition() function. Let us use 2007.



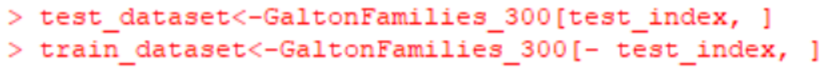
```
> library(tidyverse)
> library(caret)
> set.seed(2007)
> test_index<-createDataPartition(y, times = 1, p = 0.5, list = FALSE)
```

… (2.16)

Let us explain the roles of the arguments of the createDataPartition() function.

- The times argument tells how many random samples of indices we want to create. Here, the times argument is 1, which means we need only one such sample. … (2.17)
- Argument p tells the proportion of the indices of the data. Here, p is 0.5 means we need 50% of the data. … (2.18)
- The argument list simply tells whether we need the output of the function as a list or not. If the list argument is set to TRUE, the output will be a list; here, we set it to FALSE, so the output will not be a list. … (2.19)

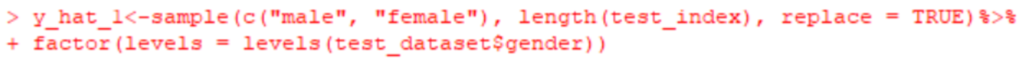
Now, using (2.16), we can obtain training and testing datasets using the following R code given as follows.



```
> test_dataset<-GaltonFamilies_300[test_index, ]
> train_dataset<-GaltonFamilies_300[- test_index, ]
```

… (2.20)

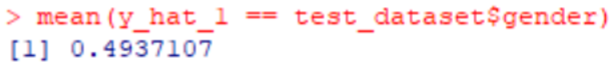
Let us first use the sample() function to decide sex of the child. We know that sample() function randomly selects a male or a female. So, sample() function is not using the feature childHeight in its prediction. It is a totally random selection. This can be done as follows.



```
> y_hat_1<-sample(c("male", "female"), length(test_index), replace = TRUE)%>%
+ factor(levels = levels(test_dataset$gender))
```

… (2.21)

Here, the target variable is gender, which is a categorical variable. We know that for a categorical variable, the suitable measure that can be used to check how many of them are predicted correctly is the proportion (you may refer to page number 30 of the course MST-019). This can be seen by running the following code. … (2.22)

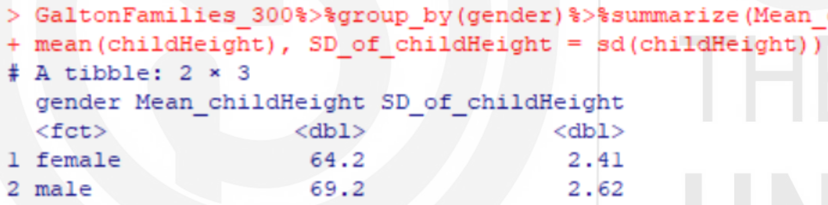


```
> mean(y_hat_1 == test_dataset$gender)
[1] 0.4937107
```

… (2.23)

The result is close to 50% as expected, because we selected males and females randomly. This output may change run to run because we did not use any seed before running the R code (2.21). Note that here we have compared the values of the data object y_hat_1 with the values of the data object test_dataset$gender, not with train_dataset$gender, because evaluation of an algorithm is done using the test dataset, not the training dataset. The second point is that we created the y_hat_1 variable of length equal to the length of the variable test_index (you may refer to the code mentioned in 2.21).

But we can do better if we use the information available in the dataset GaltonFamilies_300. Let us obtain some information about this dataset using two statistical tools: the mean and standard deviation of male and female children's height separately. This can be obtained by running the following R code.



```
> GaltonFamilies_300%>%group_by(gender)%>%summarize(Mean_childHeight =
+ mean(childHeight), SD_of_childHeight = sd(childHeight))
# A tibble: 2 × 3
  gender Mean_childHeight SD_of_childHeight
  <fct>             <dbl>             <dbl>
1 female             64.2              2.41
2 male               69.2              2.62
```

… (2.24)

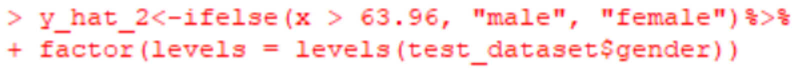
After getting this information, think for a while about how we can use this information to make a better prediction of the sex of the children.

Hope you have given some thought. One point that is striking in my mind is that the mean height of male children is 5 $(= 69.2 - 64.2)$ inches more than the mean height of female children. So, we should utilise this information in our prediction. If you have any other brilliant idea(s), then you should also give it(these) a try.

We know that the height of a population follows a normal distribution. We also know that in a normal distribution, 95% population lies within two standard deviations. But we take more than that, and consider all children having height greater than the mean minus two standard deviations as male.

So, $\text{mean} - 2(\text{Standard deviation}) = 69.2 - 2 \times 2.62 = 69.2 - 5.24 = 63.96$

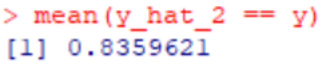
Let us predict the gender of a child as male if the height is greater than 63.96 and female otherwise. This can be done by using the following R code.



```
> y_hat_2<-ifelse(x > 63.96, "male", "female")%>%
+ factor(levels = levels(test_dataset$gender))
```

… (2.25)

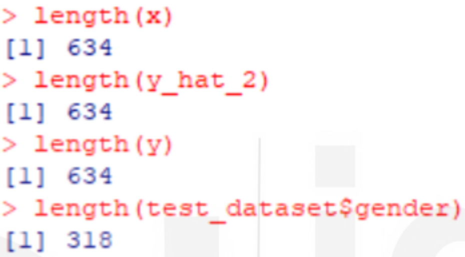
Let us now check the overall accuracy.



```
> mean(y_hat_2 == y)
[1] 0.8359621
```

… (2.26)

Great! Overall accuracy has increased from 49.37% to 83.59%. It indicates that there may be some other cutoff value which can do better. We will try that. But at this point, one question that may arise in your mind is here (means in 2.26), we compared y_hat_2 with y, while in (2.23) we did a comparison of y_hat_1 with test_dataset$gender, not with y. Why did we do so? Good question. It shows your learning sincerity and hunger to dive deeper into the subject. The answer to this question is y_hat_2 is created using x (you may refer to 2.25), which is of length 634, and so the length of y_hat_2 is also 634. But the length of test_dataset$gender is 318. We can verify it using the following codes.

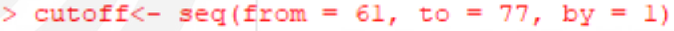


```
> length(x)
[1] 634
> length(y_hat_2)
[1] 634
> length(y)
[1] 634
> length(test_dataset$gender)
[1] 318
```

We know that in a normal distribution, within 3 standard deviations around the mean more than 99% values lie. So, let us obtain these values as follows.

$$\text{mean} \mp 3(\text{Standard deviation}) = 69.2 \mp 3 \times 2.62 = 69.2 \mp 7.86$$
$$= 61.34 \text{ and } 77.06.$$

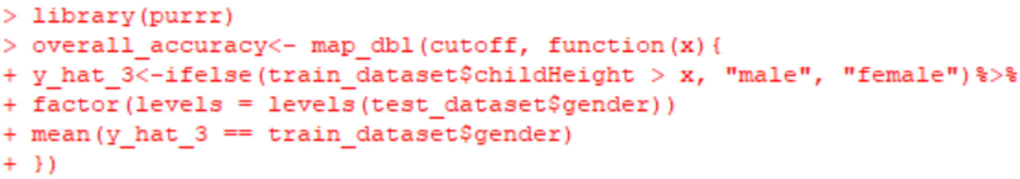
So, we check for cutoff values 61, 62, 63, …, 77. First, we create a sequence of these numbers as follows and save it under the name of the cutoff variable.



```
> cutoff<- seq(from = 61, to = 77, by = 1)
```

… (2.27)

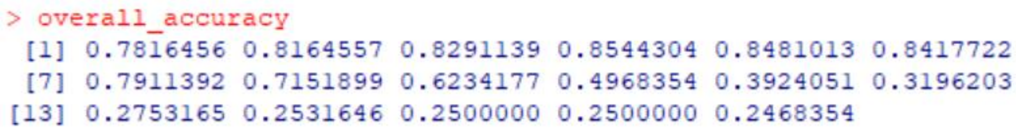
Now, we want to apply (2.25) for each value of the variable cutoff. That is, we want to apply a function to different values of an already created variable. Such a function that can perform this kind of job is the map_dbl() function of the purrr package. First, you have to install this package and then load it before applying the map_dbl() function. I have already installed it in my system. So, I only need to load it.



```
> library(purrr)
> overall_accuracy<- map_dbl(cutoff, function(x){
+ y_hat_3<-ifelse(train_dataset$childHeight > x, "male", "female")%>%
+ factor(levels = levels(test_dataset$gender))
+ mean(y_hat_3 == train_dataset$gender)
+ })
```

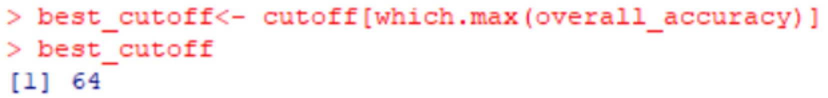
… (2.28)

Let us see all values of overall accuracy just saved in the variable overall_accuracy.



```
> overall_accuracy
 [1] 0.7816456 0.8164557 0.8291139 0.8544304 0.8481013 0.8417722
 [7] 0.7911392 0.7151899 0.6234177 0.4968354 0.3924051 0.3196203
[13] 0.2753165 0.2531646 0.2500000 0.2500000 0.2468354
```

We see that the maximum value among these is 0.8544304. Let us find out at which value of cutoff it is obtained. This can be found out by running the following code.

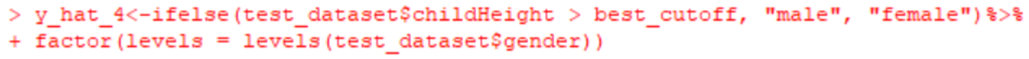


```
> best_cutoff<- cutoff[which.max(overall_accuracy)]
> best_cutoff
[1] 64
```

… (2.29)

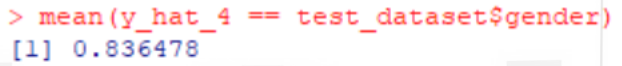
We see that among the whole numbers 61, 62, 63, …, 77, the number 64 plays the role of the best cutoff. If we go in decimal numbers, then there may be some other number better than 64. If you are interested in getting that number, then you have to just replace the value of ‘by’ argument in (2.27) by 0.1 or 0.01 or 0.001, etc. and run the codes mentioned in (2.28) and (2.29), you will get the number of your interest.

Note that in (2.28), we have used the training data set, while, as mentioned in (2.4), we know that evaluation of a machine learning algorithm is done using a testing dataset instead of the training dataset. So, let us do that as follows.



```
> y_hat_4<-ifelse(test_dataset$childHeight > best_cutoff, "male", "female")%>%
+ factor(levels = levels(test_dataset$gender))
```

Overall accuracy on the test dataset can be obtained as follows.



```
> mean(y_hat_4 == test_dataset$gender)
[1] 0.836478
```

… (2.30)

It is less than the overall accuracy (0.8544304) of the training dataset but much greater than the overall accuracy (0.4937107) of the guessing model.

Now, we move toward the learning side of the point that we discussed in (2.11) and (2.12). The point of concern with overall accuracy, especially when we have a biased dataset, is that it may hide the truth with respect to the actual prediction of each category (in the present case, individual accuracy of male and female children) of the variable of bias (in the present case gender is the variable of bias because we have more male than female children). So, our next goal is to obtain the accuracy of male and female children, individually. This point and some other evaluation metrics are discussed in the next section. … (2.31)

ignou
THE PEOPLE'S
UNIVERSITY

Now, you can try the following Self-Assessment Question.

SAQ 2

Write any one line of R code where the pipe operator of the tidyverse package is used. After getting the output explain how pipe operator works. Objective of this SAQ is to tell you the silent power of pipe operator so that it can motivate you to apply pipe operator in your codes. Make a habit of applying pipe operator where possible It will make you smart in coding.

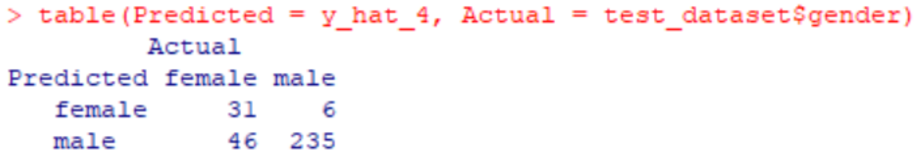
2.4 CONFUSION MATRIX AND EVALUATION METRICS

In this section, we have to continue the discussion of the previous section. Keeping our objective in view, we need four entries:

- Actually, female and machine learning algorithm also predicted female.
- Actually, female, but the machine learning algorithm predicted as male.

- Actually, male and machine learning algorithm also predicted male.
- Actually, male, but the machine learning algorithm predicted as female.

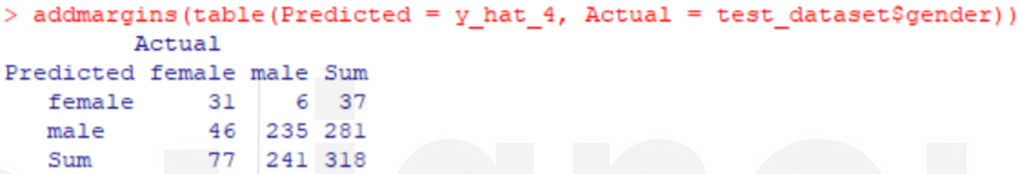
When these four entries are taken together in a 2 by 2 matrix form constitute what is known as a confusion matrix. So, the confusion matrix for the dataset discussed in the previous section can be obtained by running the following R code. The screenshot with output is shown as follows.



```
> table(Predicted = y_hat_4, Actual = test_dataset$gender)
         Actual
Predicted female male
   female     31    6
   male       46  235
```

(2.32)

Next, let us also obtain the margins sum of the above confusion matrix given by (2.32) by running the following R code.



```
> addmargins(table(Predicted = y_hat_4, Actual = test_dataset$gender))
         Actual
Predicted female male Sum
   female     31    6  37
   male       46  235 281
   Sum        77  241 318
```

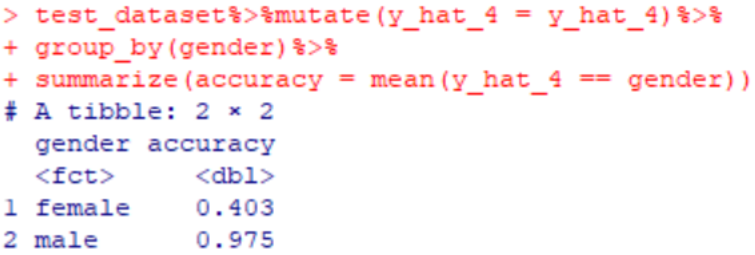
… (2.33)

We see that actual male children are 241 and actual female children are 77. Also, the number of male children who are actually males and also predicted by the machine learning algorithm as male is 235. Similarly, the number of female children who are actually female and also predicted female is 31. So,

the accuracy of predicting male children correctly $\frac{235}{241} \simeq 0.9751$ and

the accuracy of predicting female children correctly $\frac{31}{77} \simeq 0.4026$

Using R code, we can obtain the same as follows.



```
> test_dataset%>%mutate(y_hat_4 = y_hat_4)%>%
+ group_by(gender)%>%
+ summarize(accuracy = mean(y_hat_4 == gender))
# A tibble: 2 × 2
  gender accuracy
  <fct>     <dbl>
1 female    0.403
2 male      0.975
```

You have noted the huge difference in the individual accuracy of male and female children 97.5% versus 40.3% respectively, while overall accuracy was 83.6% (you may refer to 2.30). This was the point I wanted to highlight: you should keep in mind when you compute overall accuracy, especially when you are working with a biased dataset. This happened because the prevalence of male children is much higher than the prevalence of female children in the GaltonFamilies_300 dataset. My objective in creating a biased dataset, GaltonFamilies_300, from the original dataset, GaltonFamilies, was to give you an exposure to this important learning that you should have as a data scientist. … (2.34)

As promised in (2.30), in this section, other than the confusion matrix, we will also learn some other evaluation metrics which can also be used to notice the issues because of the presence of the problem of a higher prevalence of one category compared to the other category. Two such measures are sensitivity and specificity. Recall that you have already studied sensitivity and specificity in Unit 4 of the course MST-019. To refresh your memory, let us copy and paste Table 4.1 of the course MST-019 as follows.

Table 4.1 Result of Screening Test and Disease Status

		Disease Status		
		Yes (D^+)	**No (D^-)**	**Total**
Screening test result	T^+	a (True positive)	b (False positive)	a + b (Total number of participants who are tested positive by the test)
	T^-	c (False negative)	d (True negative)	c+ d (Total number of participants who are tested negative by the test)
Total		a + c (Total screened participants who actually have disease)	b + d (Total screened participants who actually do not have disease)	a + b +c + d = N (Total number of screened participants)

Let us prepare a similar table by following the terminology of the present course on machine learning instead of epidemiology that was discussed in Block-1 of the Course MST-019. After doing that, Table 4.1 will look like this, and we give it the name Table 2.1.

Table 2.1 General Result of Algorithm Prediction and Actual Status

		Actual Status		
		Actual Positive	**Actual Negative**	**Total**
Algorithm Prediction	**Predicted Positive**	a True Positive (TP)	b False Positive (FP)	a + b The total of those who are predicted positive by the algorithm
	Predicted Negative	c False Negative (FN)	d True Negative (TN)	c + d The total of those who are predicted negative by the algorithm
Total		a + c The total of those who are actually positive	b + d The total of those who are actually negative	a + b + c + d = N Grand Total

Now, we can define sensitivity and specificity as follows.

Sensitivity: The ability of an algorithm to correctly predict a positive outcome when it is actually positive. So, sensitivity is the probability, and in the terminology of the above table, it is defined as follows.

$$\text{Sensitivity} = \frac{TP}{TP + FN} = \frac{a}{a + c} \qquad \dots (2.35)$$

Remark: Sensitivity is also known as the **True Positive Rate** (TPR). It has one more name **recall**.

Specificity: The ability of an algorithm to correctly predict a negative outcome when it is actually negative. So, specificity is the probability, and in the terminology of the above table, it is defined as follows.

$$\text{Specificity} = \frac{TN}{TN + FP} = \frac{d}{b + d} \qquad \dots (2.36)$$

Remark: Specificity is also known as the **True Negative Rate** (TNR).

Recall from Unit 4 of the course MST-019: in such a situation, we define two additional measures, the positive predictive value (PPV) and the negative predictive value (NPV), as follows.

Positive Predictive Value: The proportion of outcomes that are predicted positive by the algorithm that are actually positive. In the terminology of the above table, it is defined as follows.

$$\text{Positive predictive value} = \frac{TP}{TP + FP} = \frac{a}{a + b} \qquad \dots (2.37)$$

Remark: Positive predictive value is also known as **precision**.

Negative Predictive Value: The proportion of outcomes that are predicted negative by the algorithm that are actually negative. In the terminology of the above table, it is defined as follows.

$$\text{Negative predictive value} = \frac{TN}{TN + FN} = \frac{d}{c + d} \qquad \dots (2.38)$$

Let us add two more measures to our list of measures, known as balanced accuracy and F_1-score which are defined as follows.

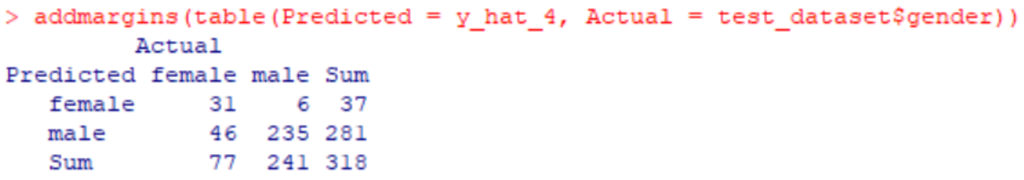
Balanced accuracy is defined as the average of sensitivity and specificity, so

$$\text{Balanced accuracy} = \frac{\text{Sensitivity} + \text{Specificity}}{2} \qquad \dots (2.39)$$

F_1-score is the harmonic mean of recall and precision, and so can be defined as follows.

$$F_1\text{-score} = 2\frac{(\text{Precision}) \times (\text{Recall})}{\text{Precision} + \text{Recall}} \qquad \dots (2.40)$$

Let us obtain these measures using the confusion matrix given by (2.33) as follows.



```
> addmargins(table(Predicted = y_hat_4, Actual = test_dataset$gender))
         Actual
Predicted female male Sum
   female     31    6  37
   male       46  235 281
   Sum        77  241 318
```

$$\text{Sensitivity (or Recall)} = \frac{TP}{TP + FN} = \frac{a}{a + c} = \frac{31}{77} \simeq 0.4026 = 40.26\%$$

$$\text{Specificity} = \frac{TN}{TN + FP} = \frac{d}{b + d} = \frac{235}{241} \simeq 0.9751 = 95.51\%$$

$$\text{Positive predictive value (or Precision)} = \frac{TP}{TP + FP} = \frac{a}{a + b} = \frac{31}{37} \simeq 0.8378$$

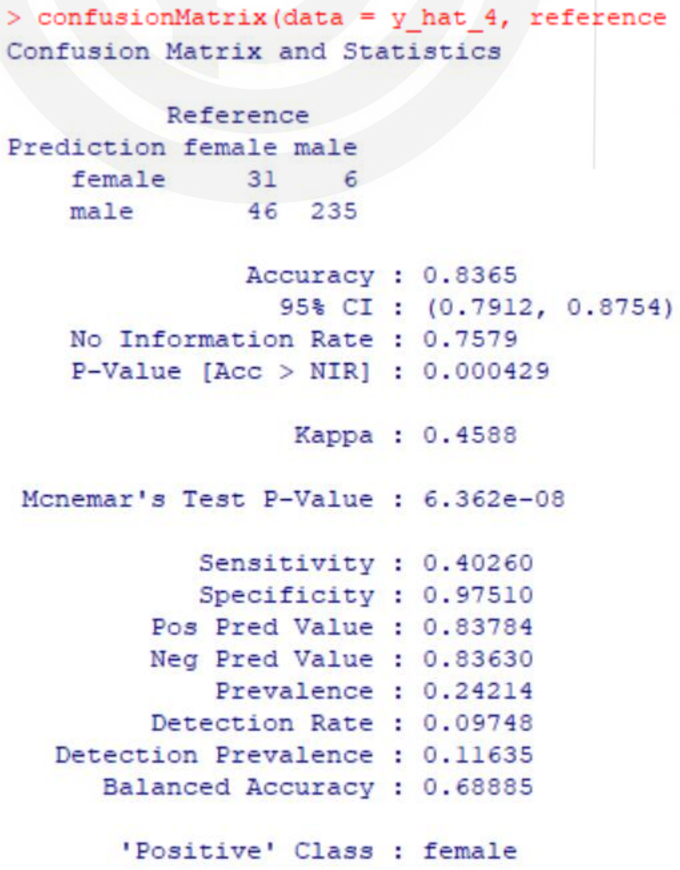
$$= 83.78\%$$

$$\text{Negative predictive value} = \frac{TN}{TN + FN} = \frac{d}{c + d} = \frac{235}{281} \simeq 0.8363 = 83.63\%$$

$$\text{Balanced accuracy} = \frac{0.4026 + 0.9751}{2} = \frac{1.3777}{2} = 0.68885 = 68.885\%$$

$$F_1\text{-score} = 2\frac{(\text{Precision}) \times (\text{Recall})}{\text{Precision} + \text{Recall}} = 2\frac{0.8378 \times 0.4026}{0.8378 + 0.4026} \simeq 0.5439$$

All these measures and some more can be obtained by using confusionMatrix() function of caret package as follows.



```
> confusionMatrix(data = y_hat_4, reference = test_dataset$gender)
Confusion Matrix and Statistics

          Reference
Prediction female male
    female     31    6
    male       46  235

               Accuracy : 0.8365
                 95% CI : (0.7912, 0.8754)
    No Information Rate : 0.7579
    P-Value [Acc > NIR] : 0.000429

                  Kappa : 0.4588

 Mcnemar's Test P-Value : 6.362e-08

            Sensitivity : 0.40260
            Specificity : 0.97510
         Pos Pred Value : 0.83784
         Neg Pred Value : 0.83630
             Prevalence : 0.24214
         Detection Rate : 0.09748
   Detection Prevalence : 0.11635
      Balanced Accuracy : 0.68885

       'Positive' Class : female
```

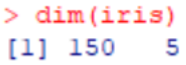
… (2.41)

ignou
THE PEOPLE'S
UNIVERSITY

Let us do one example.

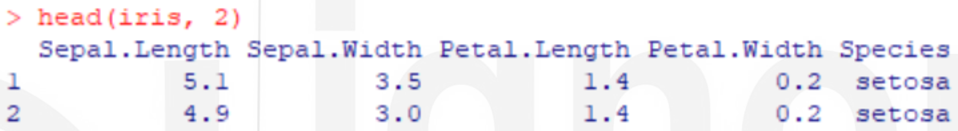
Example 1: Consider the iris dataset in R. First, add a categorical column “Setosa” which takes the yes category if the Species column of that row has the entry ‘setosa’ and no otherwise and call this new dataset “iris_setosa”. After that, split this new dataset randomly into 70:30, resulting in train_set and test_set. Train a logistic regression model with Setosa as a target variable while Sepal.Length and Sepal.Width as features. Construct a confusion matrix corresponding to this model. Hence, find sensitivity, specificity, overall accuracy, PPV, NPV, balanced accuracy and F_1-score.

Solution: We know that there are 150 rows and 5 columns in the iris dataset. This can be checked as follows.



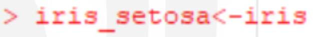
```
> dim(iris)
[1] 150   5
```

The first two rows of the dataset iris can be seen as follows.



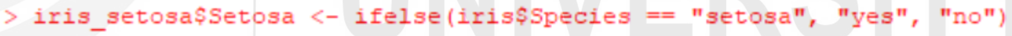
```
> head(iris, 2)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
```

Let us first make a copy of the iris dataset and give it a new name iris_setosa. This can be done as follows.



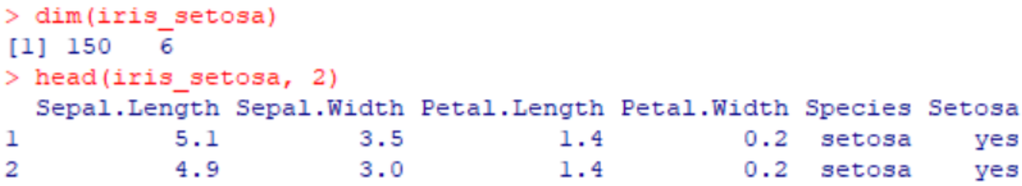
```
> iris_setosa<-iris
```

Now, in this new dataset iris_setosa, let us create a new categorical column “Setosa” which takes the ‘yes’ category if the Species column of that row has the entry ‘setosa’ and ‘no’ otherwise. This can be done as follows.



```
> iris_setosa$Setosa <- ifelse(iris$Species == "setosa", "yes", "no")
```

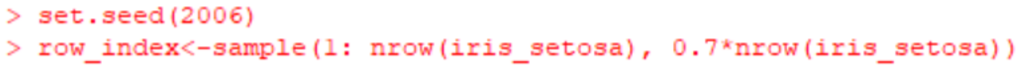
Now, one column “Setosa” has been added to the original dataset iris. This can be seen as follows.



```
> dim(iris_setosa)
[1] 150   6
> head(iris_setosa, 2)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species Setosa
1          5.1         3.5          1.4         0.2  setosa    yes
2          4.9         3.0          1.4         0.2  setosa    yes
```

Note that now we have 6 columns instead of 5 in new dataset iris_Setosa_col.

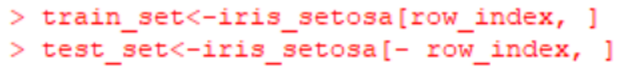
Next, we have to select a random sample of 70% of row indices of this dataset, which can be done as follows. For reproducibility purposes, we will set 2006 as the seed.



```
> set.seed(2006)
> row_index<-sample(1: nrow(iris_setosa), 0.7*nrow(iris_setosa))
```

Now, we can obtain train and test sets as follows.

ignou
THE PEOPLE'S
UNIVERSITY



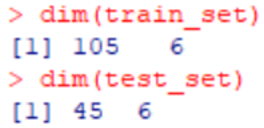
```
> train_set<-iris_setosa[row_index, ]
> test_set<-iris_setosa[- row_index, ]
```

… (2.42)

We know that 70% of 150 rows $= \frac{70}{100} \times 150 = 105$ rows and

30% of 150 rows $= \frac{30}{100} \times 150 = 45$ rows

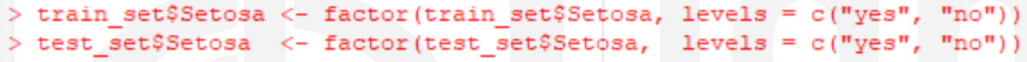
So, we expect that the number of rows in the train_set and test_set datasets should be 105 and 45, respectively. This can be verified as follows.



```
> dim(train_set)
[1] 105   6
> dim(test_set)
[1] 45  6
```

Verified. Good.

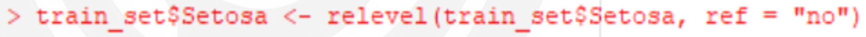
To ensure consistent factor levels where yes remains first and no remains second, we have to run the following R code.



```
> train_set$Setosa <- factor(train_set$Setosa, levels = c("yes", "no"))
> test_set$Setosa  <- factor(test_set$Setosa,  levels = c("yes", "no"))
```

… (2.43)

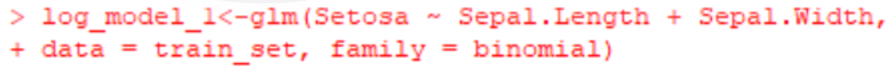
We know that the logistic model predicts probabilities. We want that model to predict P(yes) = probability of yes. We also know that glm() function predicts the probability of the second level. Here, the second level is no. So, we have to relevel. This can be done as follows.



```
> train_set$Setosa <- relevel(train_set$Setosa, ref = "no")
```

… (2.44)

Now, we can train a logistic regression model with Setosa as a target variable while Sepal.Length and Sepal.Width as features as follows.

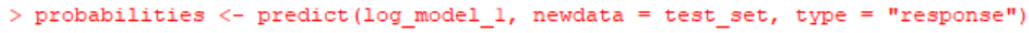


```
> log_model_1<-glm(Setosa ~ Sepal.Length + Sepal.Width,
+ data = train_set, family = binomial)
```

… (2.45)

You may get some warning messages. Ignore them.

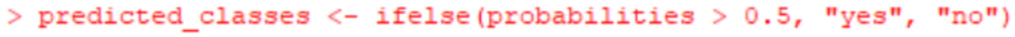
Next, we need to obtain the model's predicted probabilities. Let us store these probabilities in the variable probabilities. It can be done as follows.



```
> probabilities <- predict(log_model_1, newdata = test_set, type = "response")
```

… (2.46)

To convert these probabilities into outcomes, we have to apply a classification function. This can be done as follows.

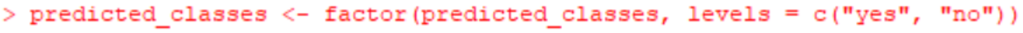


```
> predicted_classes <- ifelse(probabilities > 0.5, "yes", "no")
```

… (2.47)

ignou
THE PEOPLE'S
UNIVERSITY

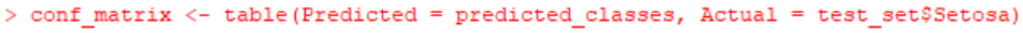
Next, before obtaining the confusion matrix, we have to convert predicted_classes into a factor variable with levels yes and no. This can be done as follows.



```
> predicted_classes <- factor(predicted_classes, levels = c("yes", "no"))
```

… (2.48)

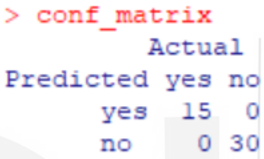
Finally, we can obtain a confusion matrix by running the following code. Let us save the output in the variable conf_matrix.



```
> conf_matrix <- table(Predicted = predicted_classes, Actual = test_set$Setosa)
```

… (2.49)

Now, the confusion matrix has been saved in the object conf_matrix, which can be seen by typing conf_matrix in the R console and hitting enter. The screenshot of the output is shown as follows.



```
> conf_matrix
         Actual
Predicted yes no
      yes  15  0
      no    0 30
```

… (2.50)

Now, if we compare the confusion matrix given by (2.50) with the general form of the confusion matrix given by (2.40), we have

$$\text{TP}=a=15,\ \text{FP}=b=0,\ \text{FN}=c=0,\ \text{TN}=d=30 \quad \dots (2.51)$$

Now, required evaluation metrics are given by

$$\text{Sensitivity (or Recall)} = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{a}{a+c} = \frac{15}{15+0} = \frac{15}{15} = 1 = 100\%$$

$$\text{Specificity} = \frac{\text{TN}}{\text{TN}+\text{FP}} = \frac{d}{b+d} = \frac{30}{0+30} = \frac{30}{30} = 1 = 100\%$$

$$\text{Overall accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{FP}+\text{FN}+\text{TN}} = \frac{a+d}{a+b+c+d} = \frac{15+30}{15+0+0+30}$$

$$= \frac{45}{45} = 1 = 100\%$$

$$\text{Positive predictive value (or Precision)} = \frac{\text{TP}}{\text{TP}+\text{FP}} = \frac{a}{a+b} = \frac{15}{15+0} = \frac{15}{15}$$

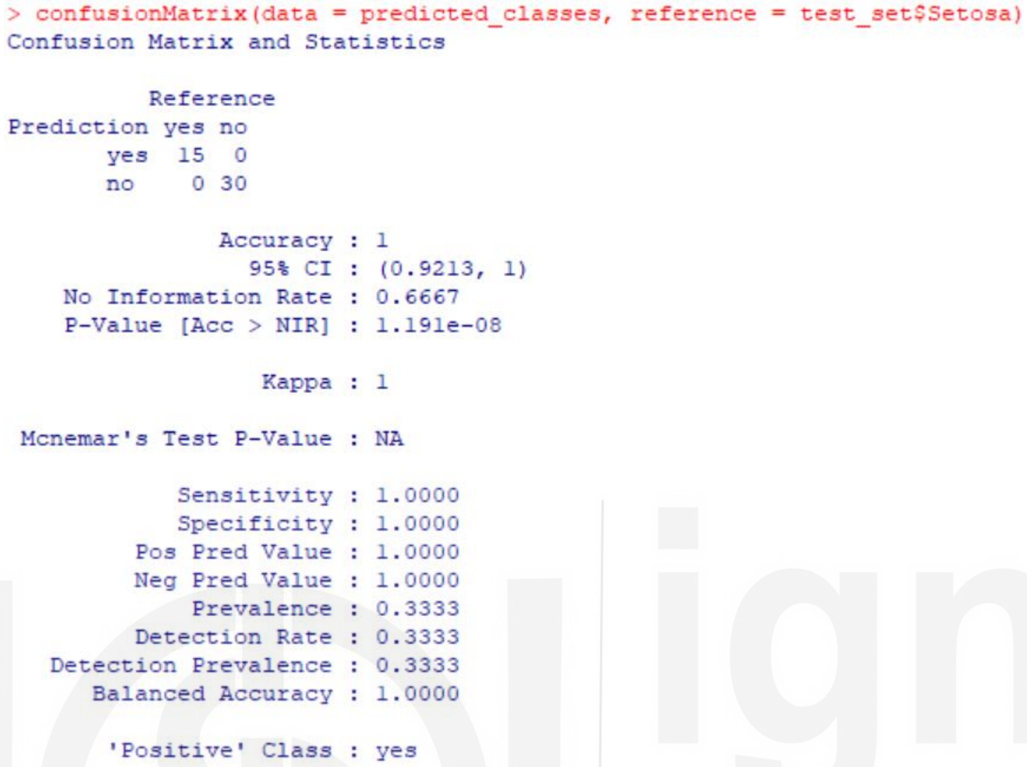
$$= 1 = 100\%$$

$$\text{Negative predictive value} = \frac{\text{TN}}{\text{TN}+\text{FN}} = \frac{d}{c+d} = \frac{30}{0+30} = \frac{30}{30} = 1 = 100\%$$

$$\text{Balanced accuracy} = \frac{\text{Sensitivity}+\text{Specificity}}{2} = \frac{100+100}{2}\% = \frac{200}{2}\% = 100\%$$

$$F_1\text{-score} = 2\frac{(\text{Precision})\times(\text{Recall})}{\text{Precision} + \text{Recall}} = 2\frac{100\times 100}{100+100}\% = 100\%$$

All these measures and some more can be obtained by using confusionMatrix() function of caret package as follows.



```
> confusionMatrix(data = predicted_classes, reference = test_set$Setosa)
Confusion Matrix and Statistics

          Reference
Prediction yes no
       yes  15  0
       no    0 30

               Accuracy : 1
                 95% CI : (0.9213, 1)
    No Information Rate : 0.6667
    P-Value [Acc > NIR] : 1.191e-08

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.3333
         Detection Rate : 0.3333
   Detection Prevalence : 0.3333
      Balanced Accuracy : 1.0000

       'Positive' Class : yes
```

Now, you can try the following Self-Assessment Question.

SAQ 3

Re do Example 1, just this time make an even split as train and test datasets. That is, make 50 : 50 split instead of 70 : 30 split. Do not obtain evaluation metrics manually like we did in Example 1. I assume that you can make a good practice of that from the TEE point of view as per your need. Like Example 1, just obtain those evaluation metrics using the confusionMatrix() function of the caret package. Comment on the output of the confusionMatrix() function by comparing the output with the output of Example 1.

2.5 SUMMARY

A brief summary of what we have covered in this unit is given as follows:

- **Training Dataset**: The original dataset is partitioned into two datasets generally in the ratio 80 : 20 or 75 : 25 or 70 : 30. That is, the larger dataset contains 70 to 80% of the original dataset. This larger dataset, which is a random subset of the original dataset, is used to train the machine learning algorithm and is known as training dataset.
- **Testing Dataset**: The original dataset is partitioned into two datasets generally in the ratio 80 : 20 or 75 : 25 or 70 : 30. That is, the smaller dataset contains 20 to 30% of the original dataset. This smaller dataset, which is a random subset of the original dataset, is used to test the trained machine learning algorithm and is known as the test dataset.

ignou
THE PEOPLE'S
UNIVERSITY

- If the algorithm performs well in the training dataset but poorly in the testing dataset, it is an indication of the problem of **overfitting**.
- **Sensitivity or True Positive Rate (TPR) or Recall**: The ability of an algorithm to correctly predict a positive outcome when it is actually positive. So, sensitivity is the probability that is defined as follows:
  $$\text{Sensitivity} = \frac{TP}{TP + FN} = \frac{a}{a + c}.$$
- **Specificity or True Negative Rate (TNR)**: The ability of an algorithm to correctly predict a negative outcome when it is actually negative. So, specificity is the probability that is defined as follows:
  $$\text{Specificity} = \frac{TN}{TN + FP} = \frac{d}{b + d}$$
- **Positive Predictive Value or Precision**: The proportion of outcomes that are predicted positive by the algorithm that are actually positive. So, positive predictive value is the probability and is defined as follows:
  $$\text{Positive predictive value or Precision} = \frac{TP}{TP + FP} = \frac{a}{a + b}$$
- **Negative Predictive Value**: The proportion of outcomes that are predicted as negative by the algorithm that are actually negative. So, negative predictive value is the probability and is defined as follows.
  $$\text{Negative predictive value} = \frac{TN}{TN + FN} = \frac{d}{c + d}$$
- **Balanced accuracy** is defined as the average of sensitivity and specificity, so $\text{Balanced accuracy} = \dfrac{\text{Sensitivity} + \text{Specificity}}{2}$.
- F_1-score is the harmonic mean of recall and precision, and so can be defined as follows: $F_1\text{-score} = 2\dfrac{(\text{Precision}) \times (\text{Recall})}{\text{Precision} + \text{Recall}}$.

2.6 TERMINAL QUESTIONS

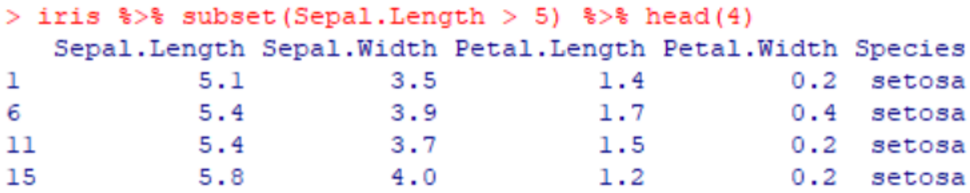
1. Build an algorithm to predict the gender of a child using the childHeight variable as a feature of the GaltonFamilies dataset in the HistData package of R by removing 200 rows randomly of female children from this dataset. Find the overall accuracy of the algorithm.
2. Obtain female and male children's overall accuracy individually and comment on the results obtained by comparing these results with the results of Sec. 2.3 of the similar model, where we removed 300 random rows of female children instead of 200 rows.

2.7 SOLUTIONS/ANSWERS

Self-Assessment Questions (SAQs)

1. It is a false statement because the training dataset is a random subset of the original dataset is ok, but it contains 70 to 80% instances of the original dataset, not 20 to 30% instances.

2. A screenshot of one sample of a line of code of R with output where the pipe operator from the tidyverse package is used is given as follows.



```
> iris %>% subset(Sepal.Length > 5) %>% head(4)
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1           5.1         3.5          1.4         0.2  setosa
6           5.4         3.9          1.7         0.4  setosa
11          5.4         3.7          1.5         0.2  setosa
15          5.8         4.0          1.2         0.2  setosa
```

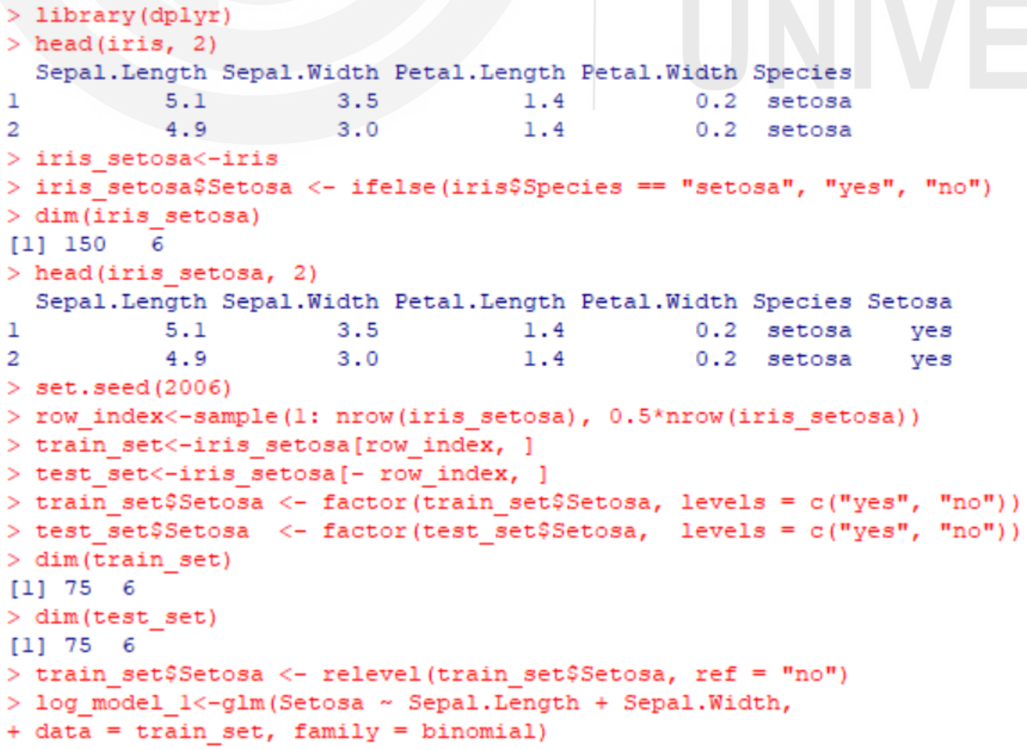
Working of pipe operator is like this:

- Whatever we have before the first pipe operator it becomes the first argument of the function just written after the first pipe operator. For example, here iris is written before the pipe operator, so it becomes the first argument of the function subset() which is written just after the first pipe operator.

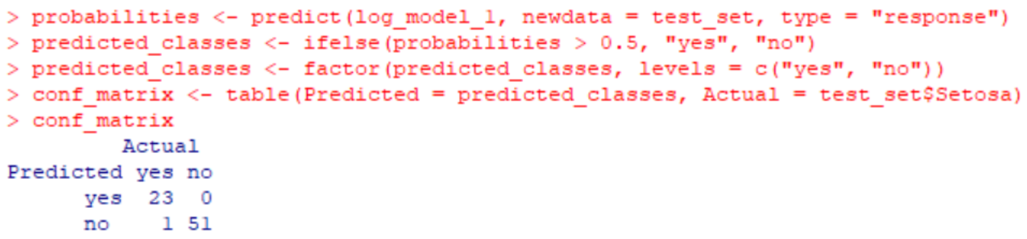
- Whatever is written before the second pipe operator it will be evaluated and its output will be the first argument of the second pipe operator R. Here first R will subset the dataset iris and select only those rows of the iris dataset where the variable Sepal.Length is greater than 5. This subset of iris dataset will be the first argument of the function head() which is written just after the second pipe operator in this line of code. Therefore final output will be the first 4 rows of the subseted iris dataset.

3. We have explained the meaning of all codes in Example 1, so here we are just running the codes and getting the required output. The screenshot of the code with output is shown as follows.

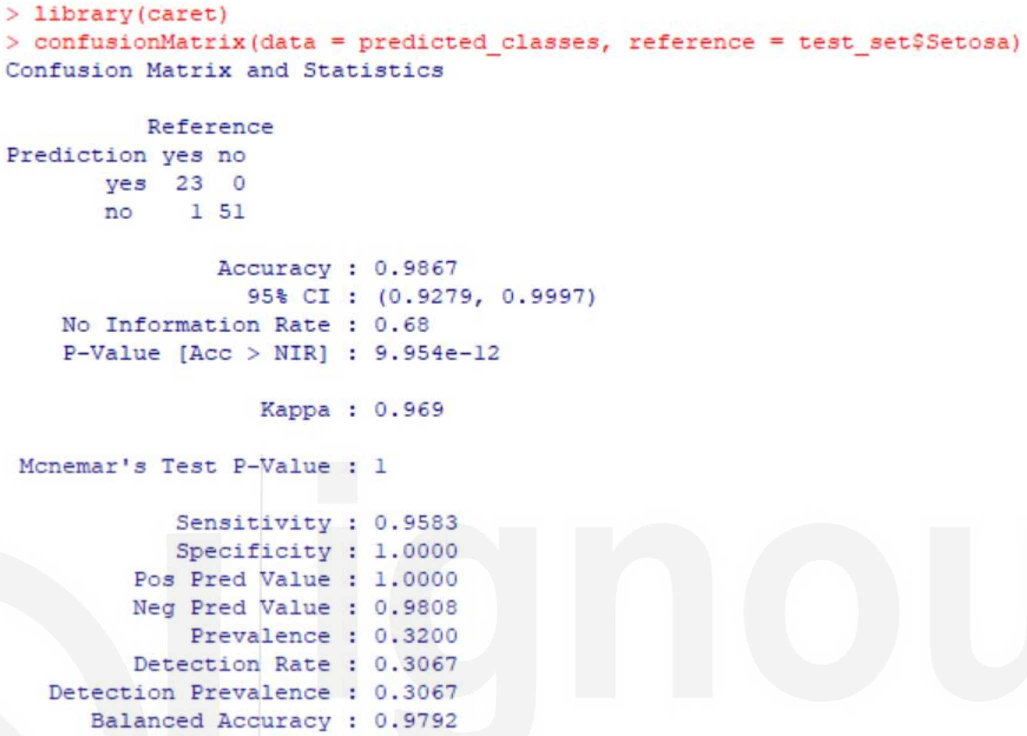
ignou
THE PEOPLE'S
UNIVERSITY



```
> library(dplyr)
> head(iris, 2)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
> iris_setosa<-iris
> iris_setosa$Setosa <- ifelse(iris$Species == "setosa", "yes", "no")
> dim(iris_setosa)
[1] 150   6
> head(iris_setosa, 2)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species Setosa
1          5.1         3.5          1.4         0.2  setosa    yes
2          4.9         3.0          1.4         0.2  setosa    yes
> set.seed(2006)
> row_index<-sample(1: nrow(iris_setosa), 0.5*nrow(iris_setosa))
> train_set<-iris_setosa[row_index, ]
> test_set<-iris_setosa[- row_index, ]
> train_set$Setosa <- factor(train_set$Setosa, levels = c("yes", "no"))
> test_set$Setosa  <- factor(test_set$Setosa,  levels = c("yes", "no"))
> dim(train_set)
[1] 75  6
> dim(test_set)
[1] 75  6
> train_set$Setosa <- relevel(train_set$Setosa, ref = "no")
> log_model_1<-glm(Setosa ~ Sepal.Length + Sepal.Width,
+ data = train_set, family = binomial)
```



```
> probabilities <- predict(log_model_1, newdata = test_set, type = "response")
> predicted_classes <- ifelse(probabilities > 0.5, "yes", "no")
> predicted_classes <- factor(predicted_classes, levels = c("yes", "no"))
> conf_matrix <- table(Predicted = predicted_classes, Actual = test_set$Setosa)
> conf_matrix
         Actual
Predicted yes no
      yes  23  0
      no    1 51
```

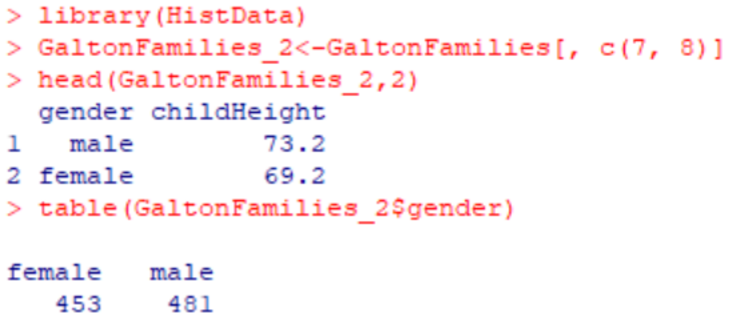


```
> library(caret)
> confusionMatrix(data = predicted_classes, reference = test_set$Setosa)
Confusion Matrix and Statistics

          Reference
Prediction yes no
       yes  23  0
       no    1 51

               Accuracy : 0.9867
                 95% CI : (0.9279, 0.9997)
    No Information Rate : 0.68
    P-Value [Acc > NIR] : 9.954e-12

                  Kappa : 0.969

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9583
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9808
             Prevalence : 0.3200
         Detection Rate : 0.3067
   Detection Prevalence : 0.3067
      Balanced Accuracy : 0.9792
```

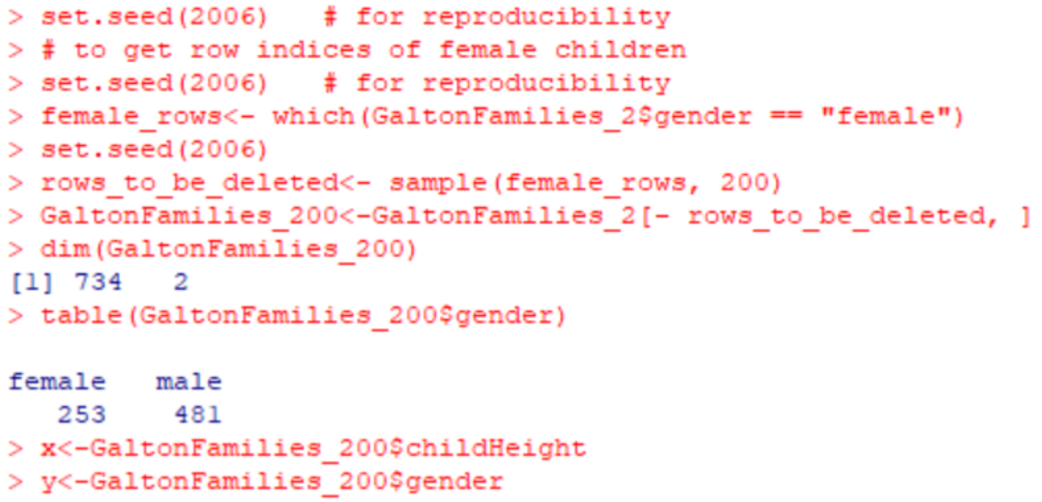
Note that this time model incorrectly classifies one case of Setosa, while all non-Setosa cases are correctly classified. So, the specificity of the model is 100% while the sensitivity of the algorithm is 23/24 = 95.83%. What can we say about the reduction of sensitivity from 100% to 95.83% compared to Example 1? One possible answer is algorithm got less data for training compared to Example 1. In Example 1 training dataset has 105 instances, while this time we have only 75 instances in the training dataset. So, we can say that the more the dataset for training, the more learning takes place.

Terminal Questions

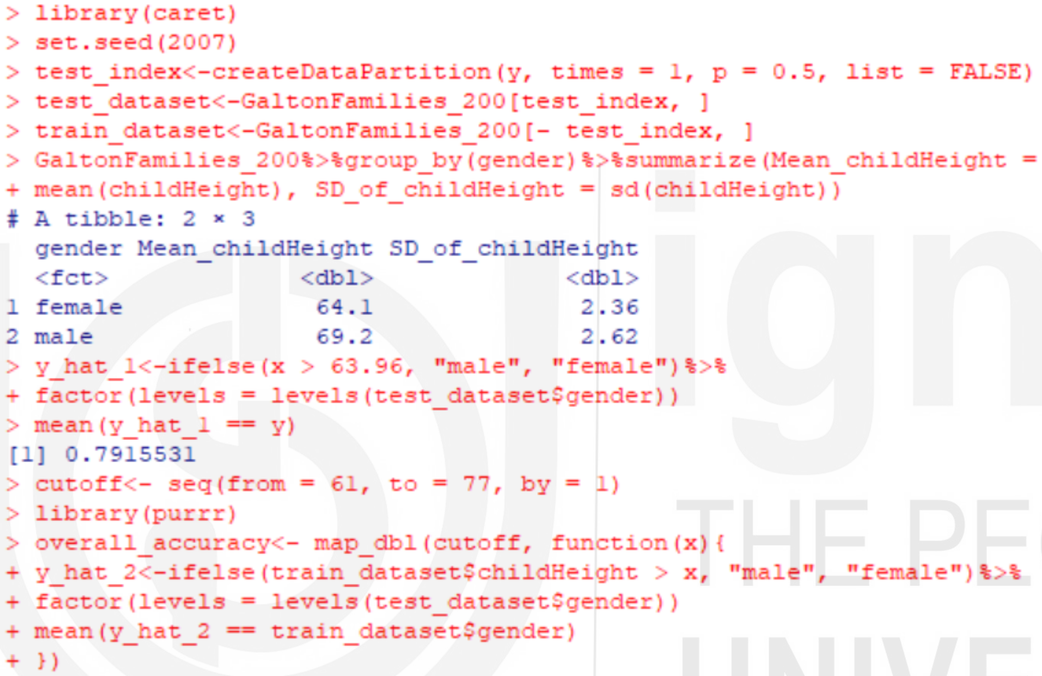
1. We have already explained all steps used in the process of building such an algorithm in Sec. 2.3. So, we are just running the final R codes on R console. Screenshot of R codes and their output is shown as follows.



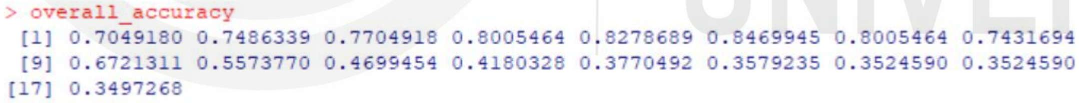
```
> library(HistData)
> GaltonFamilies_2<-GaltonFamilies[, c(7, 8)]
> head(GaltonFamilies_2,2)
  gender childHeight
1   male        73.2
2 female        69.2
> table(GaltonFamilies_2$gender)

female   male 
   453    481 
```



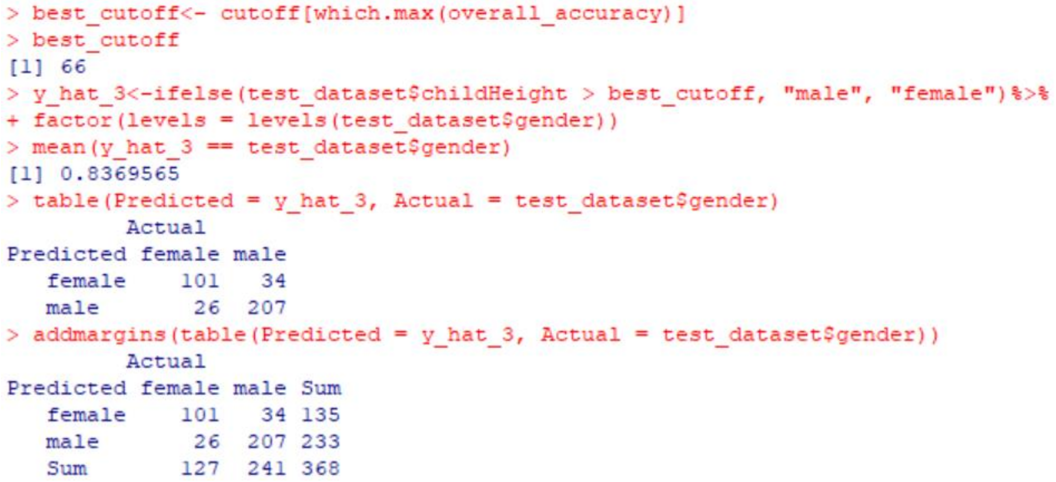
```
> set.seed(2006)   # for reproducibility
> # to get row indices of female children
> set.seed(2006)   # for reproducibility
> female_rows<- which(GaltonFamilies_2$gender == "female")
> set.seed(2006)
> rows_to_be_deleted<- sample(female_rows, 200)
> GaltonFamilies_200<-GaltonFamilies_2[- rows_to_be_deleted, ]
> dim(GaltonFamilies_200)
[1] 734   2
> table(GaltonFamilies_200$gender)

female   male 
   253    481 
> x<-GaltonFamilies_200$childHeight
> y<-GaltonFamilies_200$gender
```



```
> library(caret)
> set.seed(2007)
> test_index<-createDataPartition(y, times = 1, p = 0.5, list = FALSE)
> test_dataset<-GaltonFamilies_200[test_index, ]
> train_dataset<-GaltonFamilies_200[- test_index, ]
> GaltonFamilies_200%>%group_by(gender)%>%summarize(Mean_childHeight =
+ mean(childHeight), SD_of_childHeight = sd(childHeight))
# A tibble: 2 × 3
  gender Mean_childHeight SD_of_childHeight
  <fct>             <dbl>             <dbl>
1 female             64.1              2.36
2 male               69.2              2.62
> y_hat_1<-ifelse(x > 63.96, "male", "female")%>%
+ factor(levels = levels(test_dataset$gender))
> mean(y_hat_1 == y)
[1] 0.7915531
> cutoff<- seq(from = 61, to = 77, by = 1)
> library(purrr)
> overall_accuracy<- map_dbl(cutoff, function(x){
+ y_hat_2<-ifelse(train_dataset$childHeight > x, "male", "female")%>%
+ factor(levels = levels(test_dataset$gender))
+ mean(y_hat_2 == train_dataset$gender)
+ })
```

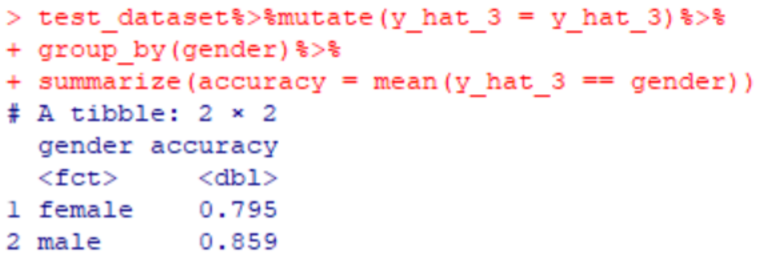


```
> overall_accuracy
 [1] 0.7049180 0.7486339 0.7704918 0.8005464 0.8278689 0.8469945 0.8005464 0.7431694
 [9] 0.6721311 0.5573770 0.4699454 0.4180328 0.3770492 0.3579235 0.3524590 0.3524590
[17] 0.3497268
```



```
> best_cutoff<- cutoff[which.max(overall_accuracy)]
> best_cutoff
[1] 66
> y_hat_3<-ifelse(test_dataset$childHeight > best_cutoff, "male", "female")%>%
+ factor(levels = levels(test_dataset$gender))
> mean(y_hat_3 == test_dataset$gender)
[1] 0.8369565
> table(Predicted = y_hat_3, Actual = test_dataset$gender)
         Actual
Predicted female male
   female    101   34
   male       26  207
> addmargins(table(Predicted = y_hat_3, Actual = test_dataset$gender))
         Actual
Predicted female male Sum
   female    101   34 135
   male       26  207 233
   Sum       127  241 368
```

2. Female and male children's overall accuracy individually can be obtained by running the following code.



```
> test_dataset%>%mutate(y_hat_3 = y_hat_3)%>%
+ group_by(gender)%>%
+ summarize(accuracy = mean(y_hat_3 == gender))
# A tibble: 2 × 2
  gender accuracy
  <fct>     <dbl>
1 female    0.795
2 male      0.859
```

Note that this time also overall accuracy of male children is more than female children but they do not have as much huge difference as they have in Sec. 2.3 where we removed 300 rows of the female children instead of 200. It means if there is more gap between prevalence of female and male children then their overall accuracy will also has more gap.

ignou
THE PEOPLE'S
UNIVERSITY

UNIT 3

ROC AND AUC

Structure

3.1 Introduction

Expected Learning Outcomes

3.2 Need of ROC

3.3 ROC Curve

3.4 Area under ROC Curve

3.5 Summary

3.6 Terminal Questions

3.7 Solutions/Answers

3.1 INTRODUCTION

In the previous unit, you studied some performance metrics of a machine learning algorithm. In this unit, we will continue that journey and discuss two more such tools known as receiver Operating Characteristic (ROC) curve and area under ROC (AUC) in Sec.s 3.3 and 3.4 respectively. In Sec. 3.2, it is discussed why we need ROC.

What we have discussed in this unit is summarised in Sec. 3.5. Self-Assessment Questions (SAQs) have been given in some sections which are generally based on the content discussed in that section. But to give you a good practice of what we have discussed in this unit some more questions based on the entire unit are given in Sec. 3.6 under the heading Terminal Questions. Due to the reason mentioned in Sec. 1.1 of Unit 1 of this course, solutions of all the SAQs and Terminal Questions are given Sec. 3.7.

In the next unit, you will study cross validation.

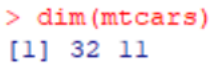
Expected Learning Outcomes

After completing this unit, you should be able to:

- answer the question why we need ROC;
- explain what is ROC and plot the same using roc() function of pROC package in R;
- explain what is AUC and obtain the same using roc() function of pROC package in R.

3.2 NEED OF ROC

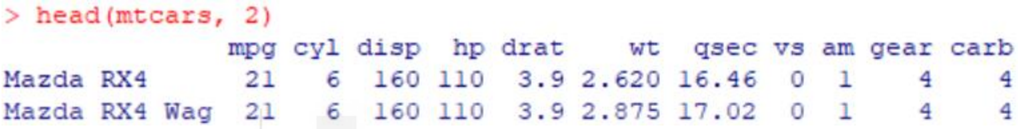
In Example 1 in Sec. 2.4 of the previous unit, we trained a logistic regression model with Setosa as a target variable while Sepal.Length and Sepal.Width as features. Similarly, to explain the need of ROC curve let us train a logistic regression model on the mtcars dataset to predict whether a car has automatic or manual transmission based on engine and performance variables: mpg, hp and wt. First of all, let us see the dimensions of this dataset by running the following R code.



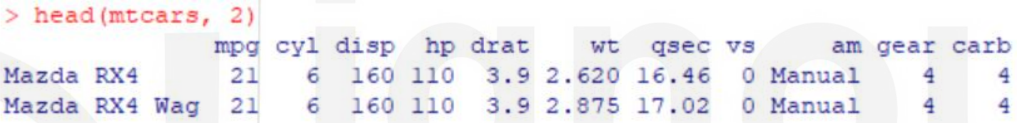
```
> dim(mtcars)
[1] 32 11
```

… (3.1)

So, we see that there are 32 rows and 11 variables in this dataset. Let us see first two rows of this dataset. This can be done as follows.

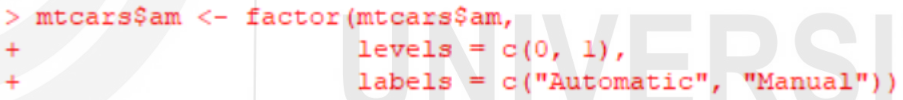


```
> head(mtcars, 2)
              mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4      21   6  160 110  3.9 2.620 16.46  0  1    4    4
Mazda RX4 Wag  21   6  160 110  3.9 2.875 17.02  0  1    4    4
```



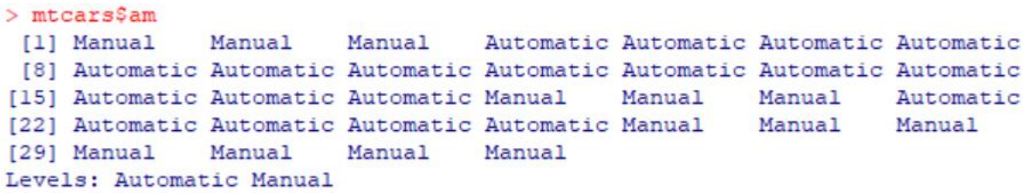
```
> head(mtcars, 2)
              mpg cyl disp  hp drat    wt  qsec vs     am gear carb
Mazda RX4      21   6  160 110  3.9 2.620 16.46  0 Manual    4    4
Mazda RX4 Wag  21   6  160 110  3.9 2.875 17.02  0 Manual    4    4
```

Here name of the target variable is ‘am’ while the names of the features are mpg, and hp. Here, target variable ‘am’ has two categories 0 and 1, where 0 indicates automatic transmission and 1 indicates a manual transmission. To make categories 0 and 1 self-explanatory, let us change the names of these categories to Automatic and Manual, respectively. This can be done as follows.



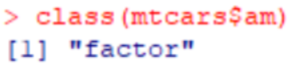
```
> mtcars$am <- factor(mtcars$am,
+                     levels = c(0, 1),
+                     labels = c("Automatic", "Manual"))
```

After doing this, let us see all the entries of the target variable ‘am’. This can be done as follows.



```
> mtcars$am
 [1] Manual    Manual    Manual    Automatic Automatic Automatic Automatic
 [8] Automatic Automatic Automatic Automatic Automatic Automatic Automatic
[15] Automatic Automatic Automatic Manual    Manual    Manual    Automatic
[22] Automatic Automatic Automatic Automatic Manual    Manual    Manual
[29] Manual    Manual    Manual    Manual
Levels: Automatic Manual
```

Let us check the class of the target variable ‘am’. This can be done as follows.



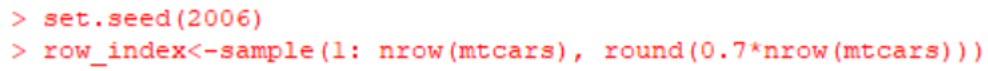
```
> class(mtcars$am)
[1] "factor"
```

… (3.2)

It is good that ‘am’ is a factor variable otherwise, we have to convert it into a factor variable.

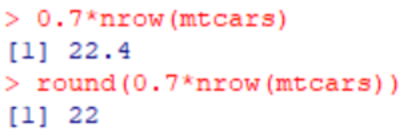
Next, we have to select a random sample of 70% of row indices of this dataset, which can be done as follows. For reproducibility purposes, we will set 2006 as the seed.

ignou
THE PEOPLE'S
UNIVERSITY



```
> set.seed(2006)
> row_index<-sample(1: nrow(mtcars), round(0.7*nrow(mtcars)))
```

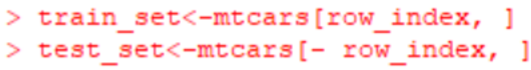
Here, the number of rows in the mtcars data frame is 32 and $0.7 \times 32 = 22.4$. So, we have used the round() function to make it a whole number. You can see the output of the following R code.



```
> 0.7*nrow(mtcars)
[1] 22.4
> round(0.7*nrow(mtcars))
[1] 22
```

… (3.3)

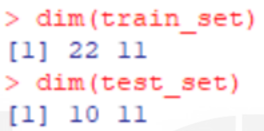
Now, we can obtain train and test sets as follows.



```
> train_set<-mtcars[row_index, ]
> test_set<-mtcars[- row_index, ]
```

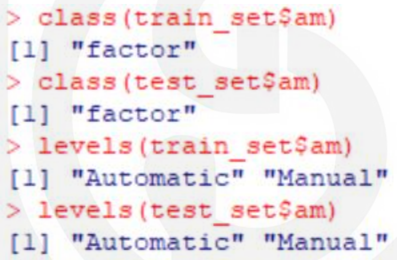
… (3.4)

We have seen that after rounding off 70% of 32 comes out 22. So, we expect that the number of rows in the train_set and test_set datasets should be 22 and 10, respectively. This can be verified as follows.



```
> dim(train_set)
[1] 22 11
> dim(test_set)
[1] 10 11
```

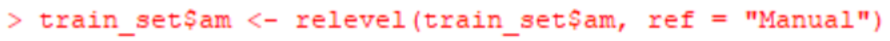
Verified. Good. Let us check the class and levels of the ‘am’ variable in the train_set and test_set datasets. This can be done as follows.



```
> class(train_set$am)
[1] "factor"
> class(test_set$am)
[1] "factor"
> levels(train_set$am)
[1] "Automatic" "Manual"
> levels(test_set$am)
[1] "Automatic" "Manual"
```

… (3.5)

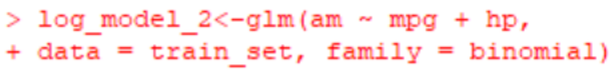
We know that the logistic model predicts probabilities. We want that model to predict P(Automatic) = probability of Automatic. We also know that glm() function predicts the probability of the second level. Here, the second level is Manual. So, we have to relevel. This can be done as follows.



```
> train_set$am <- relevel(train_set$am, ref = "Manual")
```

… (3.6)

Now, we can train a logistic regression model with ‘am’ as a target variable while mpg, and hp as features as follows.

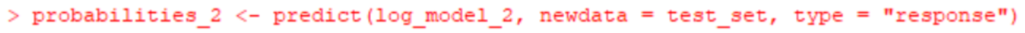


```
> log_model_2<-glm(am ~ mpg + hp,
+ data = train_set, family = binomial)
```

… (3.7)

You may get some warning messages. Ignore them.

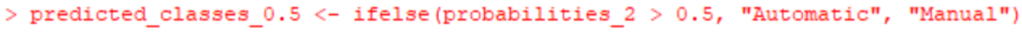
Next, we need to obtain the model's predicted probabilities. Let us store these probabilities in the variable probabilities_2. It can be done as follows.



```
> probabilities_2 <- predict(log_model_2, newdata = test_set, type = "response")
```

… (3.8)

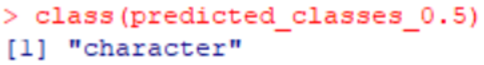
To convert these probabilities into outcomes, we have to apply a classification function. This can be done as follows.



```
> predicted_classes_0.5 <- ifelse(probabilities_2 > 0.5, "Automatic", "Manual")
```

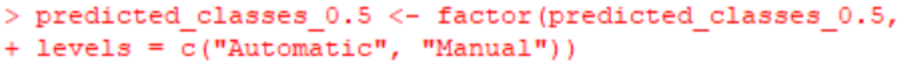
… (3.9)

Let us see the class of the predicted_classes_0.5 object.



```
> class(predicted_classes_0.5)
[1] "character"
```

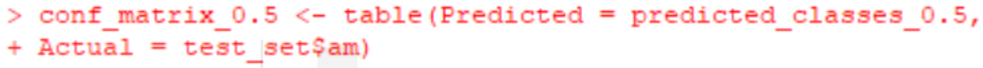
So, before obtaining the confusion matrix, we have to convert predicted_classes_0.5 into a factor variable with levels Automatic and Manual. This can be done as follows.



```
> predicted_classes_0.5 <- factor(predicted_classes_0.5,
+ levels = c("Automatic", "Manual"))
```

… (3.10)

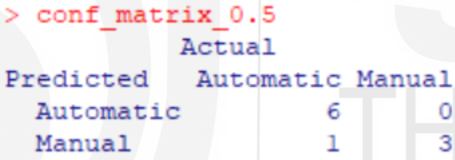
Finally, we can obtain a confusion matrix by running the following code. Let us save the output in the variable conf_matrix_0.5.



```
> conf_matrix_0.5 <- table(Predicted = predicted_classes_0.5,
+ Actual = test_set$am)
```

… (3.11)

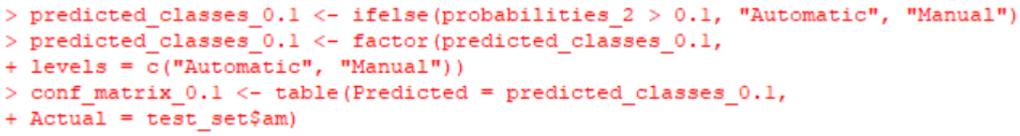
Now, the confusion matrix has been saved in the object conf_matrix_0.5, which can be seen by typing conf_matrix_0.5 in the R console and hitting enter. The screenshot of the output is shown as follows.



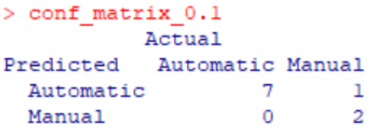
```
> conf_matrix_0.5
           Actual
Predicted   Automatic Manual
  Automatic         6      0
  Manual            1      3
```

… (3.12)

In (3.9), we have used 0.5 as the threshold value for defining the classification function. Now, let us take four more threshold values as 0.1, 0.4, 0.8 and 0.9 and see the effect of threshold value on the confusion matrix. This can be done one by one as follows.

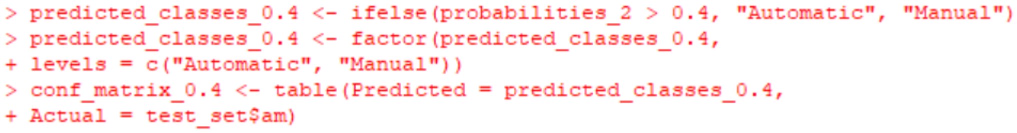


```
> predicted_classes_0.1 <- ifelse(probabilities_2 > 0.1, "Automatic", "Manual")
> predicted_classes_0.1 <- factor(predicted_classes_0.1,
+ levels = c("Automatic", "Manual"))
> conf_matrix_0.1 <- table(Predicted = predicted_classes_0.1,
+ Actual = test_set$am)
```

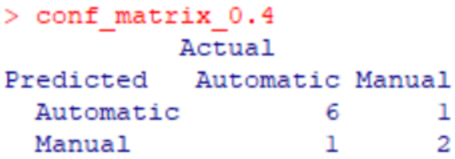


```
> conf_matrix_0.1
           Actual
Predicted   Automatic Manual
  Automatic         7      1
  Manual            0      2
```

… (3.13)



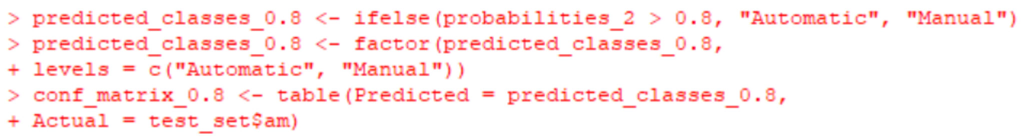
```
> predicted_classes_0.4 <- ifelse(probabilities_2 > 0.4, "Automatic", "Manual")
> predicted_classes_0.4 <- factor(predicted_classes_0.4,
+ levels = c("Automatic", "Manual"))
> conf_matrix_0.4 <- table(Predicted = predicted_classes_0.4,
+ Actual = test_set$am)
```



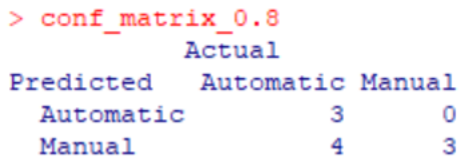
```
> conf_matrix_0.4
           Actual
Predicted   Automatic Manual
  Automatic         6      1
  Manual            1      2
```

… (3.14)

ignou
THE PEOPLE'S
UNIVERSITY

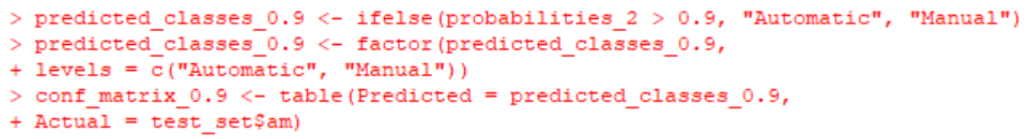


```
> predicted_classes_0.8 <- ifelse(probabilities_2 > 0.8, "Automatic", "Manual")
> predicted_classes_0.8 <- factor(predicted_classes_0.8,
+ levels = c("Automatic", "Manual"))
> conf_matrix_0.8 <- table(Predicted = predicted_classes_0.8,
+ Actual = test_set$am)
```

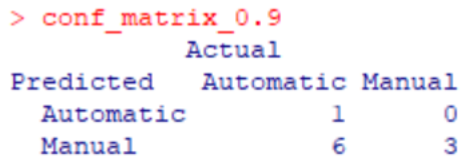


```
> conf_matrix_0.8
           Actual
Predicted   Automatic Manual
  Automatic         3      0
  Manual            4      3
```

… (3.15)



```
> predicted_classes_0.9 <- ifelse(probabilities_2 > 0.9, "Automatic", "Manual")
> predicted_classes_0.9 <- factor(predicted_classes_0.9,
+ levels = c("Automatic", "Manual"))
> conf_matrix_0.9 <- table(Predicted = predicted_classes_0.9,
+ Actual = test_set$am)
```



```
> conf_matrix_0.9
           Actual
Predicted   Automatic Manual
  Automatic         1      0
  Manual            6      3
```

… (3.16)

We note that we have different confusion matrices for different threshold values 0.1, 0.4, 0.5, 0.8 and 0.9 (you may refer to 3.12 to 3.16). If confusion matrices are different then evaluation metrics such as sensitivity, specificity, positive predictive value, negative predictive value, etc. will also vary. But the value of the threshold may take any value between 0 and 1. So, we need a tool which contains the information conveyed by so many confusion matrices. The name of such a tool is receiver operating characteristic (ROC) curve, which summarises all this information through the graph of a curve. … (3.17)

This gives the answer to the question of why we need ROC. The job of this section was to give an answer to this question. We have got the answer. So, we move to the next section where we will learn to draw the ROC curve in R programming language.

3.3 ROC CURVE

In the previous unit, you studied sensitivity and specificity (you may refer to 2.35 and 2.36). You know that to draw the graph of a curve, we need one variable on the x-axis and one variable on the y-axis. To **draw the graph of the ROC curve,** we take sensitivity on the y-axis and 1 – Specificity on the x-axis. … (3.18)

From (2.35) and (2.36) of the previous unit, you know that

$$\text{Sensitivity} = \frac{TP}{TP+FN} = \frac{a}{a+c} \qquad \ldots (3.19)$$

Sensitivity is also known as the **True Positive Rate** (TPR). It has one more name **recall**. … (3.20)

$$\text{Specificity} = \frac{TN}{TN+FP} = \frac{d}{b+d} \qquad \ldots (3.21)$$

Specificity is also known as the **True Negative Rate** (TNR). … (3.22)

$$\text{So, } 1 - \text{Specificity} = 1 - \frac{TN}{TN+FP} = \frac{TN+FP-TN}{TN+FP} = \frac{FP}{TN+FP} \qquad \ldots (3.23)$$

$$= 1 - \frac{d}{b+d} = \frac{b+d-d}{b+d} = \frac{b}{b+d} \qquad \ldots (3.24)$$

From the expression (3.23) note that numerator of **1 – Specificity** is FP (False positive), so it is also known as the **False Positive Rate** (FPR). … (3.25)

where a, b, c, d are entries of four cells of a 2 by 2 confusion matrix given as follows:

Table 3.1 General Result of Algorithm Prediction and Actual Status

		Actual Status	
		Actual Positive	**Actual Negative**
Algorithm Prediction	**Predicted Positive**	a True Positive (TP)	b False Positive (FP)
	Predicted Negative	c False Negative (FN)	d True Negative (TN)

In view of (3.19), (3.24), and Table 3.1, we can say that as entries of the confusion matrix change, then sensitivity and (1 – specificity) will change. But we know that as we change the value of the threshold in the classification function, then entries in the confusion matrix change (you may refer to 3.12 to 3.16). So, as we change the threshold value in the classification function, then sensitivity and 1 – specificity change. Consequently, as values of sensitivity and (1 – specificity) change, the ROC curve will change (you may refer to 3.18). So, as claimed in the previous section ROC curve summarises the information related to the change in the value of the threshold (you may refer to 3.17). … (3.26)

Since sensitivity and specificity are both probabilities. So, we have

$$0 \le \text{Sensitivity} \le 1 \qquad \ldots (3.27)$$

$$\text{and } 0 \le \text{Specificity} \le 1 \qquad \ldots (3.28)$$

Let us obtain a range of 1 – specificity. Rewriting (3.28), we have

$$0 \le \text{Specificity} \le 1$$

Multiplying by – 1 throughout, we get

$$\Rightarrow -0 \ge -\text{Specificity} \ge -1 \quad \left[\begin{array}{l}\text{Inequalities have changed, you may refer}\\ \text{to (6.88) of Unit 6 in the course MST-011}\end{array}\right]$$

Adding 1 throughout, we get

$$\Rightarrow 1-0 \ge 1-\text{Specificity} \ge -1+1$$

$$\Rightarrow 1 \ge 1-\text{Specificity} \ge 0$$

Or $0 \le 1 - \text{Specificity} \le 1$ … (3.29)

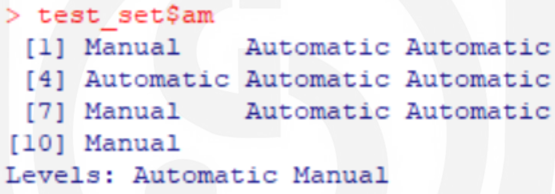
In view of (3.25), we can also say that

Or $0 \le \text{False Positive Rate} \le 1$ … (3.30)

Now, we want to draw the graph of the ROC curve. As mentioned in (3.18), to draw **the ROC curve,** we take sensitivity on the y-axis and 1 – Specificity on the x-axis. To draw the ROC curve, we have a function roc() in pROC package of R. So, it is good news that there is no need to do manual efforts to draw the graph of ROC curve. However, being a statistics learner, you should have the basic idea of the manual procedure of doing it. Keeping this in view, we will obtain the values of sensitivity and 1 – specificity for two extreme cases when the value of threshold is 0 and 1, other than the threshold values 0.1, 0.4, 0.5, 0.8 and 0.9 corresponding to which we have already obtained confusion matrices (you may refer to 3.12 to 3.16).

First extreme case when the value of the threshold is 0:

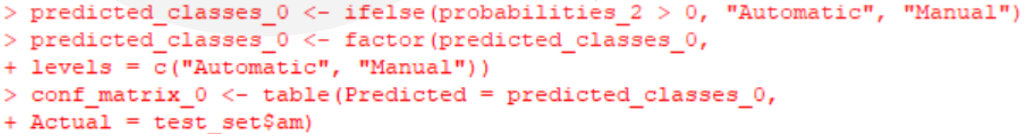
Before proceeding to obtain the confusion matrix corresponding to the threshold value 0. Let us first have a look at the variable ‘am’ in the test_set data frame to see how many Automatic and Manual cars are there. This can be seen by running the following R code.



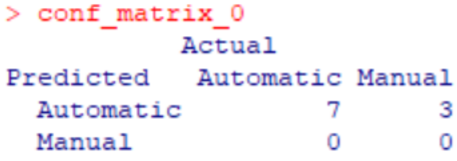
```
> test_set$am
 [1] Manual    Automatic Automatic
 [4] Automatic Automatic Automatic
 [7] Manual    Automatic Automatic
[10] Manual
Levels: Automatic Manual
```

… (3.31)

We see that out of 10 cars, 7 are Automatic and 3 are Manual. This is the truth that we know. Now, we will see what the logistic regression algorithm predicts. So, let us set the threshold to 0 and obtain the confusion matrix for this threshold.



```
> predicted_classes_0 <- ifelse(probabilities_2 > 0, "Automatic", "Manual")
> predicted_classes_0 <- factor(predicted_classes_0,
+ levels = c("Automatic", "Manual"))
> conf_matrix_0 <- table(Predicted = predicted_classes_0,
+ Actual = test_set$am)
```



```
> conf_matrix_0
            Actual
Predicted    Automatic Manual
  Automatic          7      3
  Manual             0      0
```

… (3.32)

As expected, because 0 is the threshold value, all cars should be predicted as Automatic by the logistic regression algorithm, since the probability cannot be negative. So, sensitivity (probability of predicting a positive class when it is actually positive, and this will certainly happen in this case) should be 1. Also, when the threshold is 0, then there is no scope to predict a car as Manual, so the specificity (probability of predicting a negative class when it is actually negative, but this cannot happen in this case) should be 0. This can be verified by obtaining sensitivity and specificity using (3.19), (3.21), and (3.32) as follows.

$$\text{Sensitivity} = \frac{TP}{TP+FN} = \frac{a}{a+c} = \frac{7}{7+0} = \frac{7}{7} = 1 \qquad \dots (3.33)$$

$$\text{Specificity} = \frac{TN}{TN+FP} = \frac{d}{b+d} = \frac{0}{3+0} = \frac{0}{3} = 0$$

False positive rate = 1 – Specificity $= 1 - 0 = 1$ … (3.34)

In view of (3.18), (3.33) and (3.34), one point on the ROC curve is $A(1, 1)$.

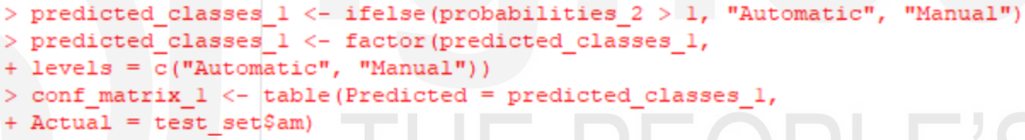
The interpretation of the point A(1, 1) is:

- y coordinate of this point is 1 implies sensitivity = 1 which means algorithm has correctly classified all the Automatic cars.
- x coordinate of this point is 1 implies 1 – specificity = 1 implies this specificity = 0 which means algorithm has incorrectly classified all the Manual cars.

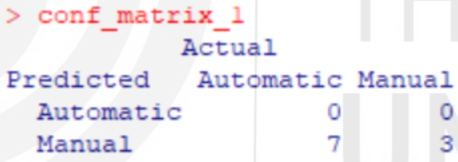
Next, we consider the second extreme case.

Second extreme case when the value of the threshold is 1:

Like the case of extreme value 0, let us take the other extreme value 1 as the threshold and obtain the confusion matrix corresponding to this threshold value. This can be done as follows.



```
> predicted_classes_1 <- ifelse(probabilities_2 > 1, "Automatic", "Manual")
> predicted_classes_1 <- factor(predicted_classes_1,
+ levels = c("Automatic", "Manual"))
> conf_matrix_1 <- table(Predicted = predicted_classes_1,
+ Actual = test_set$am)
```



```
> conf_matrix_1
           Actual
Predicted   Automatic Manual
  Automatic         0      0
  Manual            7      3
```

… (3.35)

As expected, because 1 is the threshold value, all cars should be predicted as Manual by the logistic regression algorithm, since the probability cannot be greater than 1. So, sensitivity (probability of predicting a positive class when it is actually positive, but this cannot happen in this case) should be 0. Also, when the threshold is 1, then there is no scope to predict a car as Automatic, so the specificity (probability of predicting a negative class when it is actually negative, and this will certainly happen in this case) should be 1. This can be verified by obtaining sensitivity and specificity using (3.19), (3.21), and (3.35) as follows.

$$\text{Sensitivity} = \frac{TP}{TP+FN} = \frac{a}{a+c} = \frac{0}{0+7} = \frac{0}{7} = 0 \qquad \dots (3.36)$$

$$\text{Specificity} = \frac{TN}{TN+FP} = \frac{d}{b+d} = \frac{3}{0+3} = \frac{3}{3} = 1$$

False positive rate = 1 – Specificity $= 1 - 1 = 0$ … (3.37)

In view of (3.18), (3.36) and (3.37), one more point on the ROC curve is $B(0, 0)$.

The interpretation of the point B(0, 0) is:

- y coordinate of this point is 0 implies sensitivity = 0 which means algorithm has incorrectly classified all the Automatic cars.
- x coordinate of this point is 0 implies 1 – specificity = 0 implies this specificity = 1 which means algorithm has correctly classified all the Manual cars.

Other cases when the values of the threshold are 0.1, 0.4, 0.5, 0.8 and 0.9:

We have already obtained confusion matrices corresponding to the threshold values 0.5, 0.1, 0.4, 0.8, 0.9 in (3.12) to (3.16), respectively. So, we can obtain sensitivity and 1 – Specificity for each case as follows.

For threshold value 0.1:

$$\text{Sensitivity } = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{a}{a+c} = \frac{7}{7+0} = \frac{7}{7} = 1 \qquad \ldots (3.38)$$

$$\text{Specificity } = \frac{\text{TN}}{\text{TN}+\text{FP}} = \frac{d}{b+d} = \frac{2}{1+2} = \frac{2}{3} \simeq 0.67$$

False positive rate = 1 – Specificity $\simeq 1-0.67 = 0.33$ … (3.39)

In view of (3.18), (3.38) and (3.39), one more point on the ROC curve is $C(0.33,\ 1)$.

For threshold value 0.4:

$$\text{Sensitivity } = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{a}{a+c} = \frac{6}{6+1} = \frac{6}{7} \simeq 0.86 \qquad \ldots (3.40)$$

$$\text{Specificity } = \frac{\text{TN}}{\text{TN}+\text{FP}} = \frac{d}{b+d} = \frac{2}{1+2} = \frac{2}{3} \simeq 0.67$$

False positive rate = 1 – Specificity $\simeq 1-0.67 = 0.33$ … (3.41)

In view of (3.18), (3.40) and (3.41), one more point on the ROC curve is $D(0.33,\ 0.86)$.

For threshold value 0.5:

$$\text{Sensitivity } = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{a}{a+c} = \frac{6}{6+1} = \frac{6}{7} \simeq 0.86 \qquad \ldots (3.42)$$

$$\text{Specificity } = \frac{\text{TN}}{\text{TN}+\text{FP}} = \frac{d}{b+d} = \frac{3}{0+3} = \frac{3}{3} = 1$$

False positive rate = 1 – Specificity $= 1-1 = 0$ … (3.43)

In view of (3.18), (3.42) and (3.43), one more point on the ROC curve is $E(0,\ 0.86)$.

For threshold value 0.8:

$$\text{Sensitivity} = \frac{TP}{TP+FN} = \frac{a}{a+c} = \frac{3}{3+4} = \frac{3}{7} \simeq 0.43 \qquad \ldots (3.44)$$

$$\text{Specificity} = \frac{TN}{TN+FP} = \frac{d}{b+d} = \frac{3}{0+3} = \frac{3}{3} = 1$$

False positive rate = 1 – $\text{Specificity} = 1 - 1 = 0$ … (3.45)

In view of (3.18), (3.44) and (3.45), one more point on the ROC curve is $F(0,\ 0.43)$.

For threshold value 0.9:

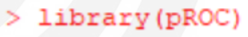
$$\text{Sensitivity} = \frac{TP}{TP+FN} = \frac{a}{a+c} = \frac{1}{1+6} = \frac{1}{7} \simeq 0.14 \qquad \ldots (3.46)$$

$$\text{Specificity} = \frac{TN}{TN+FP} = \frac{d}{b+d} = \frac{3}{0+3} = \frac{3}{3} = 1$$

False positive rate = 1 – $\text{Specificity} = 1 - 1 = 0$ … (3.47)

In view of (3.18), (3.46) and (3.47), one more point on the ROC curve is $G(0,\ 0.14)$.

Thus, we have obtained seven points (A, B, C, D, E, F, G) on the ROC curve manually. This much exposure is enough for you to understand how points that lie on ROC can be obtained. So, now we can plot ROC curve using roc() function of pROC package. For that first, you have to install pROC package in R, and then you have to load it in the current session. I have already installed this package in my system. So, I only need to load it in my current session. This can be done as follows.

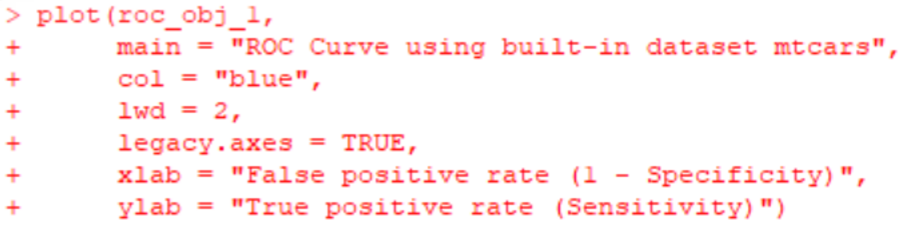


```
> library(pROC)
```

Now, ROC curve can be plotted using following R code.

```
> roc_obj_1 <- roc(test_set$am, probabilities_2)
```

… (3.48)



```
> plot(roc_obj_1,
+      main = "ROC Curve using built-in dataset mtcars",
+      col = "blue",
+      lwd = 2,
+      legacy.axes = TRUE,
+      xlab = "False positive rate (1 - Specificity)",
+      ylab = "True positive rate (Sensitivity)")
```

… (3.49)

From Fig. 3.1, note that the total plotted area for the ROC curve is a 1 by 1 square because the x-axis represents 1 – Specificity, and 1 – Specificity lies between 0 and 1. Similarly, on the y-axis, we have Sensitivity, and Sensitivity also lies between 0 and 1. So, the maximum area under the ROC curve may be 1 square unit. If the total area under the ROC curve for a model is 1, then it is considered a perfect model. The dotted diagonal line is a line where

$$\text{Sensitivity} = 1 - \text{Specificity} \qquad \ldots (3.50)$$

$$\Rightarrow \text{Sensitivity} + \text{Specificity} = 1 \qquad \ldots (3.51)$$

$$\text{Or True Positive Rate} = \text{False Positive Rate} \qquad \ldots (3.52)$$

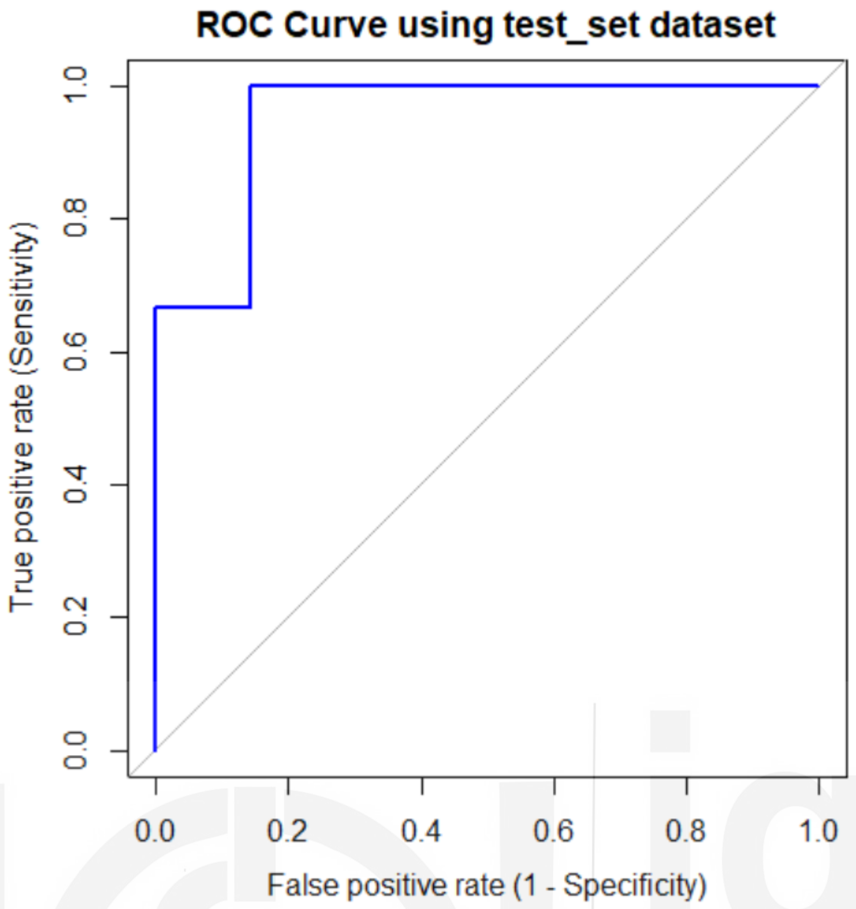


Fig. 3.1: Visualisation of ROC curve where we have considered built-in dataset mtcars with am as target variable and mpg and hp as features. Machine learning algorithm used is logistic regression

Now, you can try the following Self-Assessment Question.

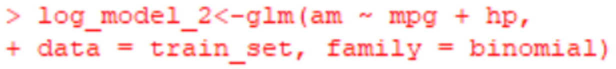
SAQ 1

Draw an ROC curve for the full dataset mtcars by considering am as target variable and mpg and hp as features. You have to apply logistic regression algorithm.

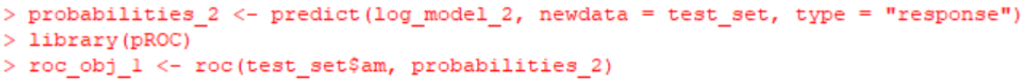
3.4 AREA UNDER ROC CURVE

In the previous section, you studied the ROC curve. You noted that ROC curve visualises the sensitivity versus 1 – Specificity for different values of threshold. In some problems, it is the top priority to classify every positive case as positive, without bothering if that results in more false positives. For example, if a plane is landed without any fault, that is not as big penalty as if a fault is not detected and plane crashes. So, in such a situation threshold should be selected in such a way that every true positive should be correctly classified as positive. Similarly, there are situations where top most priority is given to specificity instead of sensitivity. The moral of the story is keeping in view sensitivity is more important or specificity is more important ROC helps in deciding which threshold is better. … (3.53)

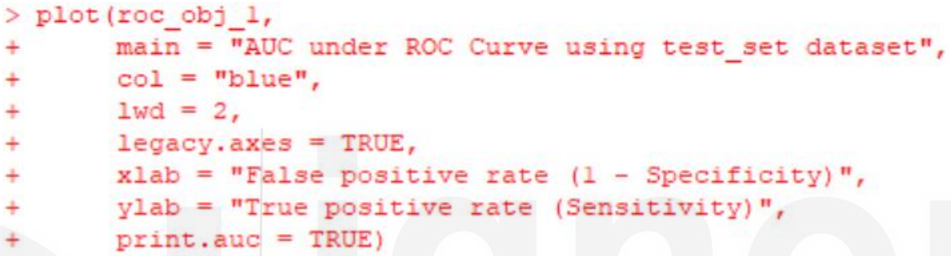
But the second point is you can apply more than one method for classification of a target variable. For example, in Sec. 3.3 of this unit you applied logistic regression algorithm. To do the same job you could also apply random forest method. Now, in deciding which method is better we obtain area under the ROC curve of the method. Usually, a method having largest area under ROC curve (AUC) is considered best. So, AUC can be used to decide which method is better than the other. In this section, we will learn to obtain AUC in R. Let us obtain AUC of the ROC plotted in Fig. 3.1 in the previous section of this unit. This can be done by running the following code. Output of these codes is shown in Fig. 3.2 where area under ROC curve is specified by the title AUC and it can be seen that its value is 0.952. … (3.54)



```
> log_model_2<-glm(am ~ mpg + hp,
+ data = train_set, family = binomial)
```



```
> probabilities_2 <- predict(log_model_2, newdata = test_set, type = "response")
> library(pROC)
> roc_obj_1 <- roc(test_set$am, probabilities_2)
```



```
> plot(roc_obj_1,
+       main = "AUC under ROC Curve using test_set dataset",
+       col = "blue",
+       lwd = 2,
+       legacy.axes = TRUE,
+       xlab = "False positive rate (1 - Specificity)",
+       ylab = "True positive rate (Sensitivity)",
+       print.auc = TRUE)
```

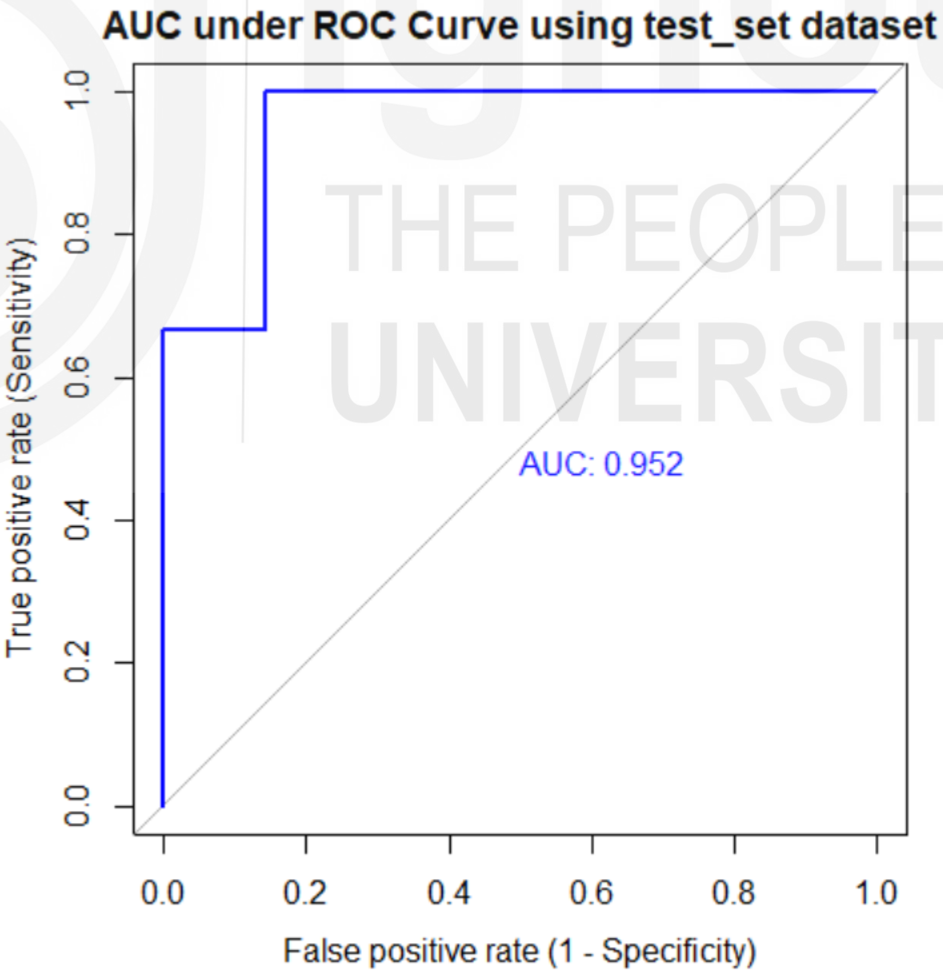


Fig. 3.2: Visualisation of the ROC curve and area under it, where we have considered the test_set dataset, which is a random test set under the set.seed(2006) of the built-in dataset mtcars with am as the target variable and mpg and hp as features. The machine learning algorithm used is logistic regression

Now, you can try the following Self-Assessment Question.

SAQ 2

Draw an ROC curve for the full dataset mtcars by considering am as target variable and mpg and hp as features. You have to apply logistic regression algorithm. Also obtain area under ROC curve.

3.5 SUMMARY

A brief summary of what we have covered in this unit is given as follows:

- **Why we need ROC Curve**: We know that we have different confusion matrices for different threshold values such as 0.1, 0.2, 0.3, 0.4, etc. If confusion matrices are different then evaluation metrics such as sensitivity, specificity, positive predictive value, negative predictive value, etc. will also vary. But the value of the threshold may take any value between 0 and 1. So, we need a tool which contains the information conveyed by so many confusion matrices. The name of such a tool is receiver operating characteristic (ROC) curve, which summarises all this information through the graph of a curve.

- **ROC Curve**: You know that to draw the graph of a curve, we need one variable on the x-axis and one variable on the y-axis. To **draw the graph of the ROC curve,** we take sensitivity on the y-axis and 1 – Specificity on the x-axis.

- **General Result of Algorithm Prediction and Actual Status**

		Actual Status	
		Actual Positive	**Actual Negative**
Algorithm Prediction	**Predicted Positive**	a True Positive (TP)	b False Positive (FP)
	Predicted Negative	c False Negative (FN)	d True Negative (TN)

- On the basis of the above table, we have
  - Sensitivity $= \dfrac{\text{TP}}{\text{TP}+\text{FN}} = \dfrac{\text{a}}{\text{a}+\text{c}}$
  - Sensitivity is also known as the **True Positive Rate** (TPR). It has one more name **recall**.
  - Specificity $= \dfrac{\text{TN}}{\text{TN}+\text{FP}} = \dfrac{\text{d}}{\text{b}+\text{d}}$
  - Specificity is also known as the **True Negative Rate** (TNR).

- So, 1 – Specificity $= 1 - \frac{TN}{TN + FP} = \frac{TN + FP - TN}{TN + FP} = \frac{FP}{TN + FP}$

$$= 1 - \frac{d}{b + d} = \frac{b + d - d}{b + d} = \frac{b}{b + d}$$

- Note that **the** numerator of **1 – Specificity** is FP (False positive), so it is also known as the **False Positive Rate** (FPR).

- **Maximum total area under ROC**: The total plotted area for the ROC curve is a 1 by 1 square because the x-axis represents 1 – Specificity, and 1 – Specificity lies between 0 and 1. Similarly, on the y-axis, we have Sensitivity, and Sensitivity also lies between 0 and 1. So, the maximum area under the ROC curve may be 1 square unit. If the total area under the ROC curve for a model is 1, then it is considered a perfect model. The dotted diagonal line in the graph of ROC curve is a line where
$\text{True Positive Rate} = \text{False Positive Rate}$

- **Area under ROC Curve (AUC)**: Obviously, you can apply more than one method for classification of a target variable. For example, in Sec. 3.3 of this unit you applied logistic regression algorithm. To do the same job you could also apply random forest method. Now, in deciding which method is better we obtain area under the ROC curve of the method. Usually, a method having largest area under ROC curve (AUC) is considered best. So, AUC can be used to decide which method is better than the other.

3.6 TERMINAL QUESTIONS

1. Consider the 'cats' dataset in MASS package in R which contains information about 144 adult cats. This dataset has three columns:
   - Column 1 is denoted by 'Sex' and represents the sex of cats
   - Column 2 is denoted by 'Bwt' and represents body weight of cats
   - Column 3 is denoted by 'Hwt' and represents heart weight of cats

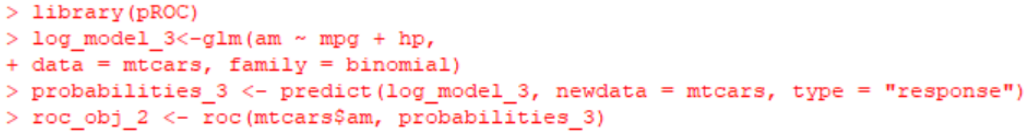
   First, randomly split the cats dataset into 70:30 as train and test sets respectively. Use last four digits of your enrolment as seed before obtaining the row indices for reproducibility purposes. Train a logistic regression model on train dataset by considering Sex as target variable and the other two variables of the dataset as features. Now, draw ROC curve for test dataset.

2. Reconsider the statement of Terminal Question 1. Obtain the area under the ROC curve drawn in Terminal Question 1.

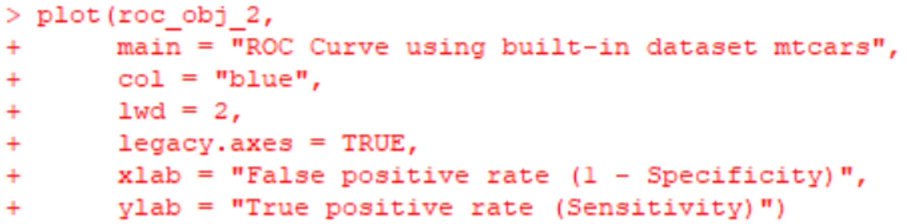
3.7 SOLUTIONS/ANSWERS

Self-Assessment Questions (SAQs)

1. We have explained the complete procedure of drawing ROC curve in R in Sec. 3.3. So, required ROC is shown in Fig. 3.3 and screenshots of R code are shown as follows.



```
> library(pROC)
> log_model_3<-glm(am ~ mpg + hp,
+ data = mtcars, family = binomial)
> probabilities_3 <- predict(log_model_3, newdata = mtcars, type = "response")
> roc_obj_2 <- roc(mtcars$am, probabilities_3)
```



```
> plot(roc_obj_2,
+       main = "ROC Curve using built-in dataset mtcars",
+       col = "blue",
+       lwd = 2,
+       legacy.axes = TRUE,
+       xlab = "False positive rate (1 - Specificity)",
+       ylab = "True positive rate (Sensitivity)")
```

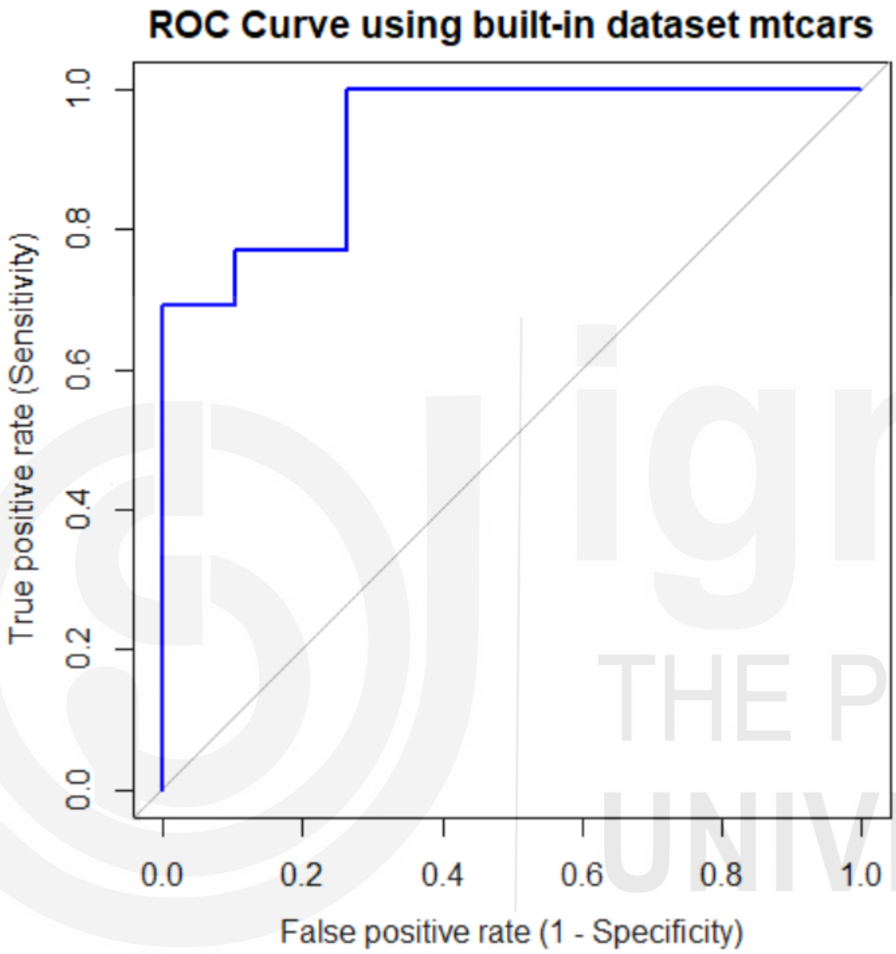
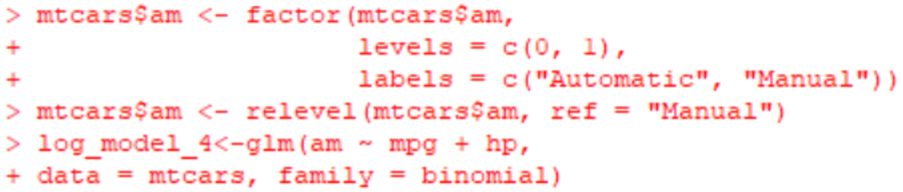
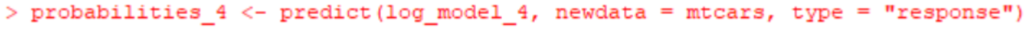


Fig. 3.3: Visualisation of ROC curve where we have considered built-in dataset mtcars with am as target variable and mpg and hp as features. Machine learning algorithm used is logistic regression

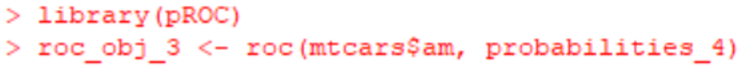
2. We have explained the complete procedure of drawing ROC curve and obtaining area under it in R in Secs. 3.3 and 3.4. So, the required ROC and AUC are shown in Fig. 3.4 and screenshots of R code are shown as follows.



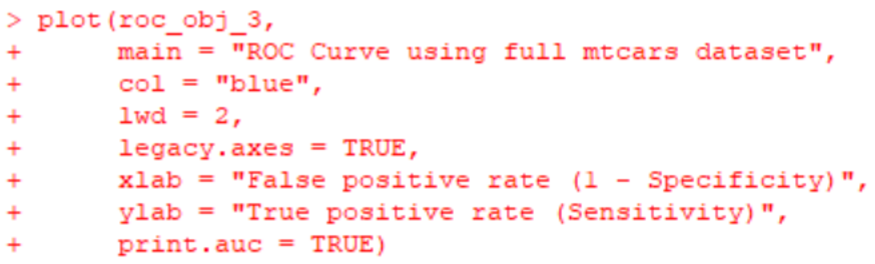
```
> mtcars$am <- factor(mtcars$am,
+                     levels = c(0, 1),
+                     labels = c("Automatic", "Manual"))
> mtcars$am <- relevel(mtcars$am, ref = "Manual")
> log_model_4<-glm(am ~ mpg + hp,
+ data = mtcars, family = binomial)
```



```
> probabilities_4 <- predict(log_model_4, newdata = mtcars, type = "response")
```



```
> library(pROC)
> roc_obj_3 <- roc(mtcars$am, probabilities_4)
```



```
> plot(roc_obj_3,
+       main = "ROC Curve using full mtcars dataset",
+       col = "blue",
+       lwd = 2,
+       legacy.axes = TRUE,
+       xlab = "False positive rate (1 - Specificity)",
+       ylab = "True positive rate (Sensitivity)",
+       print.auc = TRUE)
```

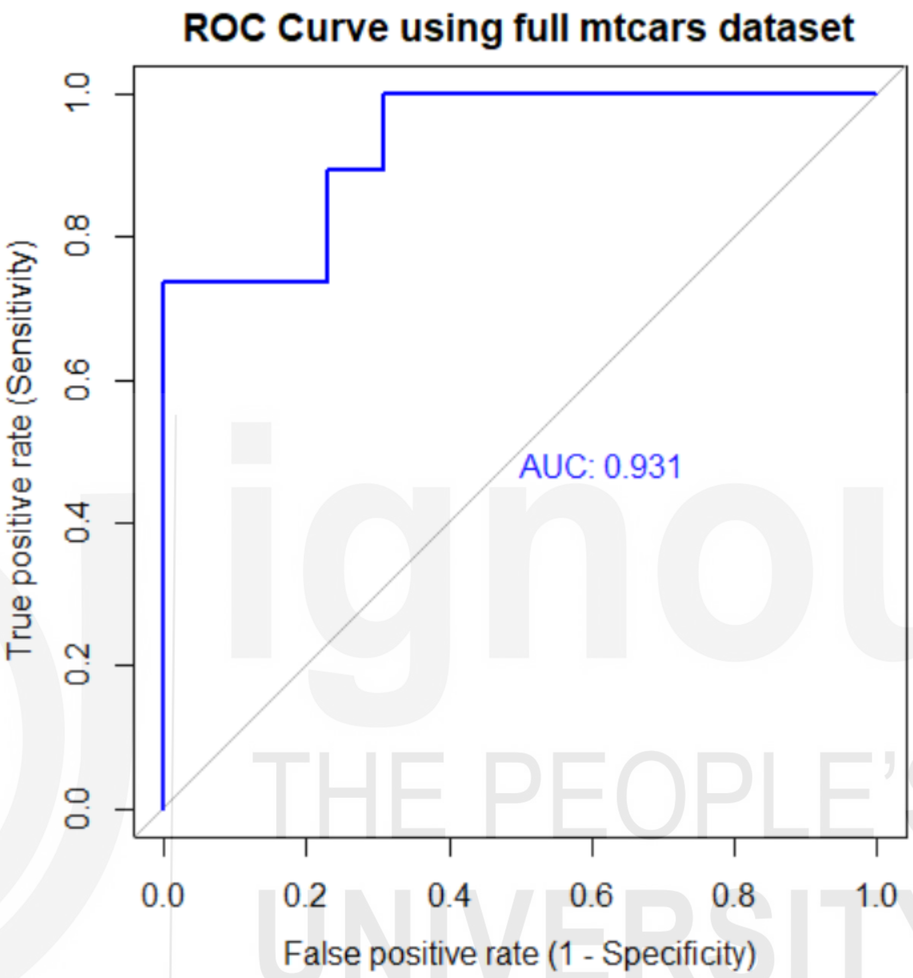
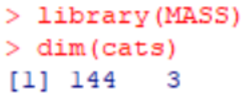


Fig. 3.4: **Visualisation of ROC curve where we have considered built-in dataset mtcars with am as target variable and mpg and hp as features. Machine learning algorithm used is logistic regression**

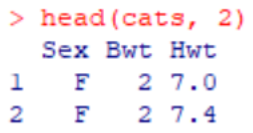
Terminal Question

1. Let us first see the dimensions of the 'cats' dataset in MASS package. This can be obtained by running the following R code.



```
> library(MASS)
> dim(cats)
[1] 144   3
```

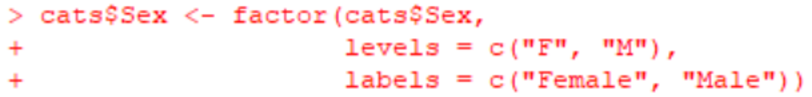
We see that there are 144 rows and 3 columns in this dataset. Next, let us see the first two rows of this dataset. This can be obtained by running the following R code.



```
> head(cats, 2)
  Sex Bwt Hwt
1   F   2 7.0
2   F   2 7.4
```

As mentioned in the statement this dataset has three columns ‘Sex’, ‘Bwt’ and ‘Hwt’.

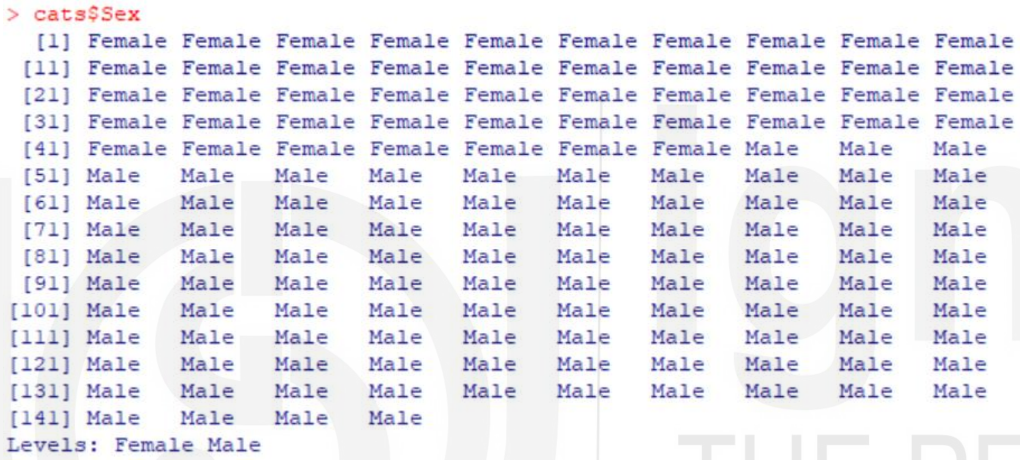
Here name of the target variable is ‘Sex’ while the names of the features are ‘Bwt’ and ‘Hwt’. Here, the target variable ‘Sex’ has two categories F and M, where F indicates female and M indicates male. To make categories 0 and 1 self-explanatory, let us change the names of these categories to Female and Male, respectively. This can be done as follows.



```
> cats$Sex <- factor(cats$Sex,
+                     levels = c("F", "M"),
+                     labels = c("Female", "Male"))
```

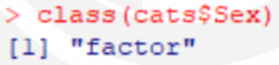
… (3.55)

After doing this, let us see all the entries of the target variable ‘Sex’. This can be done as follows.



```
> cats$Sex
  [1] Female Female Female Female Female Female Female Female Female Female
 [11] Female Female Female Female Female Female Female Female Female Female
 [21] Female Female Female Female Female Female Female Female Female Female
 [31] Female Female Female Female Female Female Female Female Female Female
 [41] Female Female Female Female Female Female Female Male   Male   Male
 [51] Male   Male   Male   Male   Male   Male   Male   Male   Male   Male
 [61] Male   Male   Male   Male   Male   Male   Male   Male   Male   Male
 [71] Male   Male   Male   Male   Male   Male   Male   Male   Male   Male
 [81] Male   Male   Male   Male   Male   Male   Male   Male   Male   Male
 [91] Male   Male   Male   Male   Male   Male   Male   Male   Male   Male
[101] Male   Male   Male   Male   Male   Male   Male   Male   Male   Male
[111] Male   Male   Male   Male   Male   Male   Male   Male   Male   Male
[121] Male   Male   Male   Male   Male   Male   Male   Male   Male   Male
[131] Male   Male   Male   Male   Male   Male   Male   Male   Male   Male
[141] Male   Male   Male   Male
Levels: Female Male
```

Let us check the class of the target variable ‘Sex’. This can be done as follows.

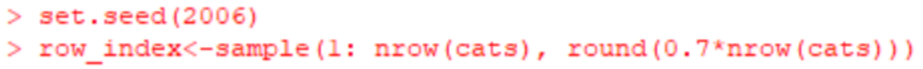


```
> class(cats$Sex)
[1] "factor"
```

… (3.56)

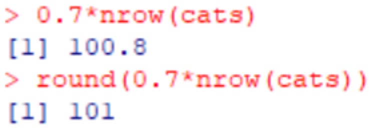
It is good that ‘Sex’ is a factor variable otherwise, we have to convert it into a factor variable.

Next, we have to select a random sample of 70% of row indices of this dataset, which can be done as follows. For reproducibility purposes, we will set 2006 as the seed.



```
> set.seed(2006)
> row_index<-sample(1: nrow(cats), round(0.7*nrow(cats)))
```

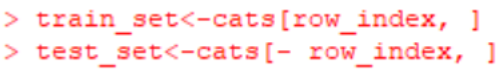
Here, the number of rows in the mtcars data frame is 144 and $0.7 \times 144 = 100.8.$ So, we have used the round() function to make it a whole number. You can see the output of the following R code.



```
> 0.7*nrow(cats)
[1] 100.8
> round(0.7*nrow(cats))
[1] 101
```

… (3.57)

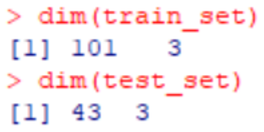
Now, we can obtain train and test sets as follows.



```
> train_set<-cats[row_index, ]
> test_set<-cats[- row_index, ]
```

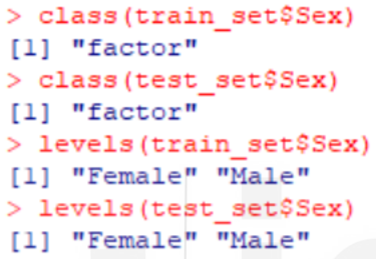
… (3.58)

We have seen that after rounding off 70% of 144 comes out 101. So, we expect that the number of rows in the train_set and test_set datasets should be 101 and 43, respectively. This can be verified as follows.



```
> dim(train_set)
[1] 101   3
> dim(test_set)
[1] 43  3
```

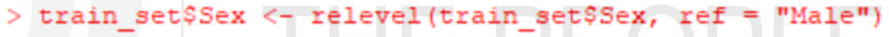
Verified. Good. Let us check the class and levels of the ‘Sex’ variable in the train_set and test_set datasets. This can be done as follows.



```
> class(train_set$Sex)
[1] "factor"
> class(test_set$Sex)
[1] "factor"
> levels(train_set$Sex)
[1] "Female" "Male"
> levels(test_set$Sex)
[1] "Female" "Male"
```

… (3.59)

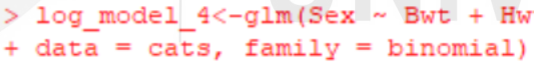
We know that the logistic model predicts probabilities. We want that model to predict P(Female) = probability of Female. We also know that glm() function predicts the probability of the second level. Here, the second level is Male. So, we have to relevel. This can be done as follows.



```
> train_set$Sex <- relevel(train_set$Sex, ref = "Male")
```

… (3.60)

Now, we can train a logistic regression model with ‘Sex’ as a target variable while Bwt, and Hwt as features as follows.

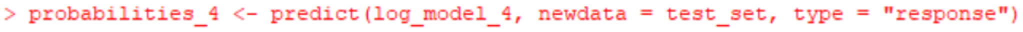


```
> log_model_4<-glm(Sex ~ Bwt + Hwt,
+ data = cats, family = binomial)
```

… (3.61)

You may get some warning messages. Ignore them.

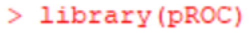
Next, we need to obtain the model's predicted probabilities. Let us store these probabilities in the variable probabilities_4. It can be done as follows.



```
> probabilities_4 <- predict(log_model_4, newdata = test_set, type = "response")
```

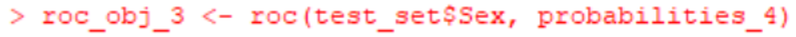
… (3.62)

So, now we can plot ROC curve using roc() function of pROC package. For that first, you have to install pROC package in R, and then you have to load it in the current session. I have already installed this package in my system. So, I only need to load it in my current session. This can be done as follows.



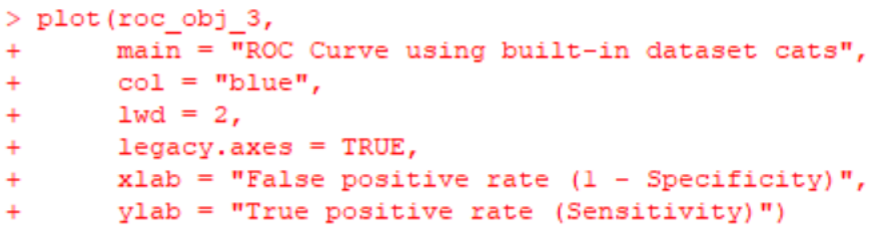
```
> library(pROC)
```

Now, the ROC curve can be plotted using the following R code.



```
> roc_obj_3 <- roc(test_set$Sex, probabilities_4)
```

ignou
THE PEOPLE'S
UNIVERSITY



```
> plot(roc_obj_3,
+       main = "ROC Curve using built-in dataset cats",
+       col = "blue",
+       lwd = 2,
+       legacy.axes = TRUE,
+       xlab = "False positive rate (1 - Specificity)",
+       ylab = "True positive rate (Sensitivity)")
```

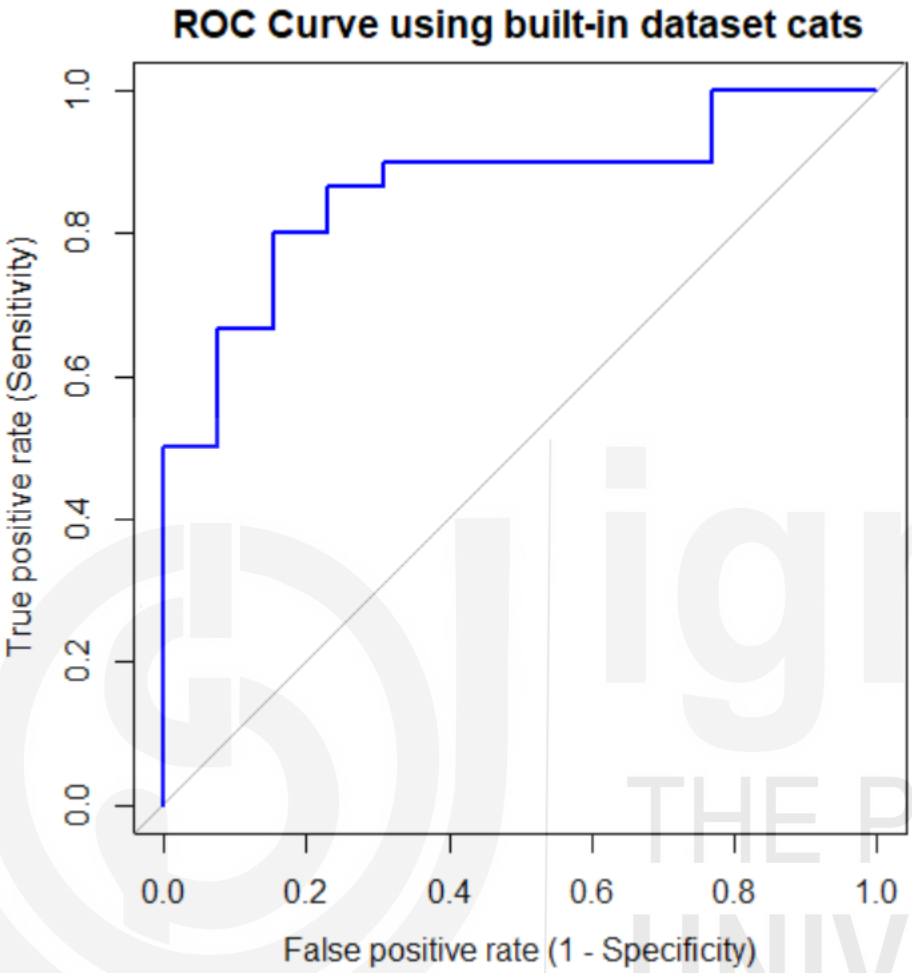
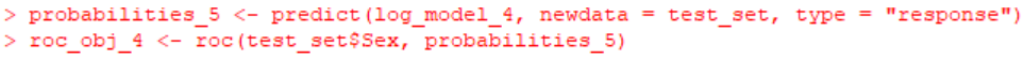
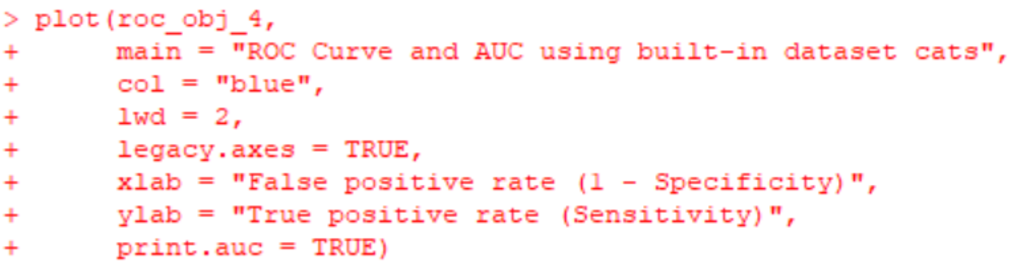


Fig. 3.5: **Visualisation of the ROC curve where we have considered the built-in dataset mtcars with am as the target variable and mpg and hp as features. The machine learning algorithm used is logistic regression**

2. The area under the ROC curve drawn in Terminal Question 1 can be obtained by specifying one more argument, print.auc = TRUE in the roc() function. The same is done as follows. Output of the code is visualised in Fig. 3.6. Make sure that before running the following code first you have to run all the codes specified in Terminal Question 1.



```
> probabilities_5 <- predict(log_model_4, newdata = test_set, type = "response")
> roc_obj_4 <- roc(test_set$Sex, probabilities_5)
```



```
> plot(roc_obj_4,
+       main = "ROC Curve and AUC using built-in dataset cats",
+       col = "blue",
+       lwd = 2,
+       legacy.axes = TRUE,
+       xlab = "False positive rate (1 - Specificity)",
+       ylab = "True positive rate (Sensitivity)",
+       print.auc = TRUE)
```

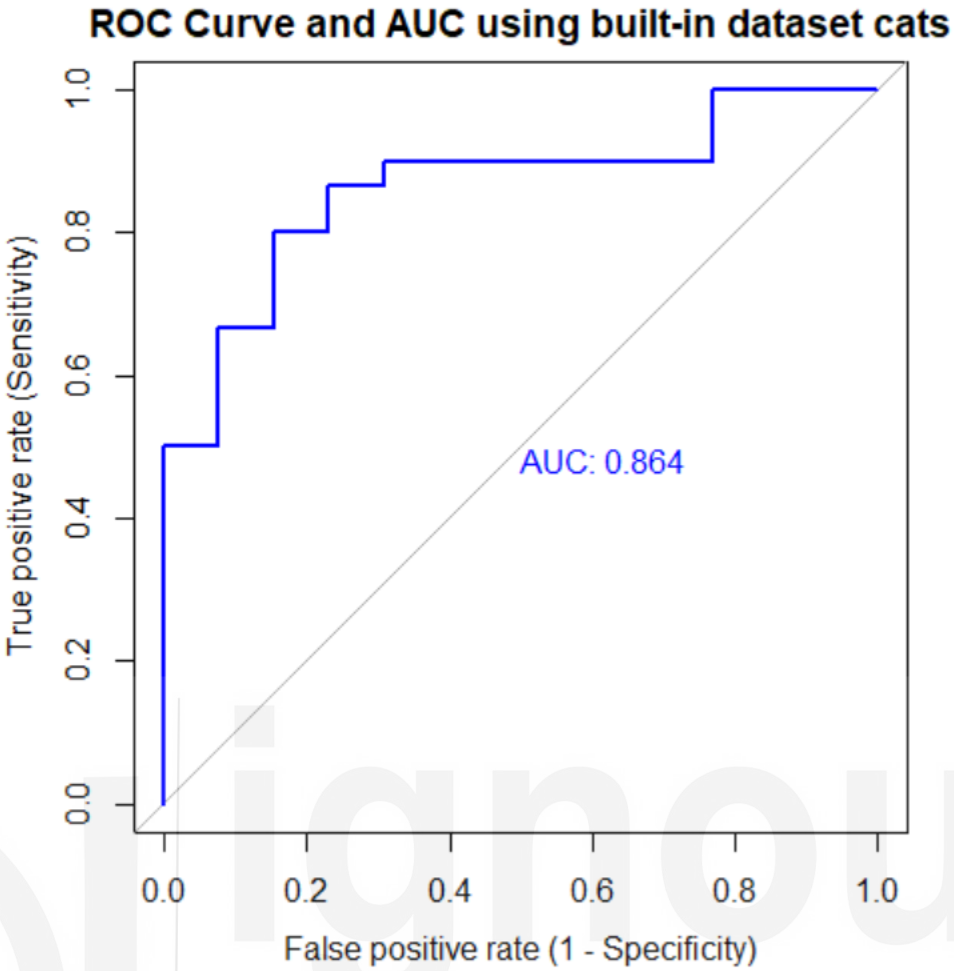


Fig. 3.6: Visualisation of ROC curve and area under this curve with the title AUC where we have considered built-in dataset cats with Sex as target variable and Bwt and Hwt as features. Machine learning algorithm used is logistic regression

ignou
THE PEOPLE'S
UNIVERSITY

UNIT 4

CROSS VALIDATION

Structure

4.1 Introduction

Expected Learning Outcomes

4.2 Definition and Need of Cross Validation

4.3 Types of Cross Validation

4.4 Cross-Validation in R

4.5 Summary

4.6 Terminal Questions

4.7 Solutions/Answers

4.1 INTRODUCTION

In Units 2 and 3, you have learnt some evaluation metrics to check which model is better than the other. In this unit, we will study one technique known as cross0validation by using which, we can improve both training and testing of the model, in particularly for small datasets. Sec. 4.2 explains the need of cross-validation. Different types of cross-validations are explained in Sec. 4.3. Sec. 4.4 explains how we can apply cross-validation in R.

What we have discussed in this unit is summarised in Sec. 4.5. Self-Assessment Questions (SAQs) have been given in some sections which are generally based on the content discussed in that section. But to give you a good practice of what we have discussed in this unit some more questions based on the entire unit are given in Sec. 4.6 under the heading Terminal Questions. Due to the reason mentioned in Sec. 1.1 of Unit 1 of this course, solutions of all the SAQs and Terminal Questions are given in Sec. 4.7.

In the next unit, you will study univariate continuous random variables.

Expected Learning Outcomes

After completing this unit, you should be able to:

- explain the need of cross-validation in machine learning algorithms;
- explain different types of cross-validation; and
- apply cross-validation in R.

4.2 DEFINITION AND OF NEED CROSS VALIDATION

In the previous two units of this course, you have seen that we partition the given dataset into two parts, known as the training dataset and the testing or test dataset. You know that our main objectives of doing that were to:

- **Train the Model on Known Data**: Training dataset is used for learning patterns, rules, and relationship from data as well as estimating parameters, and finally building the predictive model. … (4.1)
- **Evaluate Model Performance on Unseen Data**: The Model is trained using the training dataset, so the testing dataset is a new or unseen dataset for the learned model. So, the test dataset is used to check how well the trained model performs on unseen data. The ultimate goal of a machine learning model/algorithm is to make prediction on unseen data. So, using the test dataset, we can estimate the accuracy and predictive ability of the model. In short, we can say that the testing dataset acts like future real-world data to estimate how the trained model will behave. That is, the testing dataset can be treated as a reliable estimate of how the model will behave when deployed on real, future data. … (4.2)
- **To Detect Overfitting and Underfitting**: In Units 2 and 3 you have learnt to obtain many metrics to test the performance of the model. On the basis of these metrics you can identify whether the model is underfitting or overfitting explained as follows. … (4.3)
    - **Overfitting**: We say that a model is overfitting if it performs well on the training dataset but poorly on testing dataset. … (4.4)
    - **Underfitting**: We say that a model is underfitting if it performs poorly on both training and testing datasets. … (4.5)
- **To Compare or Select the Best Model**: In Unit 3, you learnt to obtain the area under the ROC curve (AUC). By calculating the AUC for the test dataset for all models under comparison, we can say which model performs well on the test dataset compared to the other models. … (4.6)

Now, to explain the need for cross-validation, let us first obtain the mean of the following five observations: 2, 5, 6, 7, 10. We know that

$$\text{Mean} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{2+5+6+7+10}{5} = \frac{30}{5} = 6 \qquad \dots (4.7)$$

Now, let us replace just one value, 10 by 1000, then new mean is given by

$$\text{Mean} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{2+5+6+7+1000}{5} = \frac{1020}{5} = 204 \qquad \dots (4.8)$$

Note that the mean has a huge effect on its value, which has increased from 6 to 204. So, inclusion of a single outlier from the given data has drastically affected the mean. … (4.9)

Now, we consider a reverse example. Suppose we have the following five observations: 9, 11, 14, 16, 1000. We know that

$$\text{Mean} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{9+11+14+16+1000}{5} = \frac{1050}{5} = 210 \qquad \ldots (4.10)$$

Now, let us replace just one value, 1000 by 10, then new mean is given by

$$\text{Mean} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{9+11+14+16+10}{5} = \frac{60}{5} = 12 \qquad \ldots (4.11)$$

Note that the mean has a huge effect on its value, which has decreased from 210 to 12. So, exclusion of a single outlier from the given data has drastically affected the mean. … (4.12)

This type of problem, due to the presence or absence of a single outlier in the testing dataset, can vastly affect the performance of the machine learning algorithm. This problem can be addressed by making more than one test dataset. This is the basic idea behind cross-validation. So, you got the answer to the question of why we need cross-validation. … (4.13)

Now, we can define cross-validation as follows.

Cross-Validation: In cross-validation, instead of partitioning the given dataset into a single training dataset and a single testing dataset, we partition the given dataset into many disjoint subsets of equal size (as far as possible) and each subset is known as a fold. After that we proceed iteratively as follows.

Iteration 1: We keep first subset/fold aside as a test dataset and consider all remaining subsets/folds (Folds 2 to last fold) as a training dataset. Obtain performance measure.

Iteration 2: We keep second subset/fold aside as a test dataset and consider all remaining subsets/folds (Fold 1, 3, 4 to last fold) as a training dataset. Obtain performance measure.

and so on.

Last Iteration: We keep last subset/fold aside as a test dataset and consider all remaining subsets/folds (Fold 1 to second last fold) as a training dataset. Obtain performance measure.

So, in this process we will get as many estimates of the performance measure on different test sets as the number of subsets/folds. After training and testing the model as many times as the number of subsets/folds, we take average of the performance measure. This process is known as cross-validation.… (4.14)

Obviously, it will take more time in training and testing the model because if we do 5 subsets of the given dataset, then we will have 5 estimates of the performance measure, one for each test dataset. A typical 5-fold cross-validation is shown in Table 4.1, where

- middle 5 columns denote 5 folds. … (4.15)

- 5 rows of the first column represent five iterations. In the first iteration, Fold 1 (20% of the total original dataset) will be considered as a test set, while Folds 2 to 5 together (80% of the total original dataset) will be considered as a train dataset. In the second iteration, Fold 2 (20% of the total original dataset) will be considered as a test set, while Folds 1 and 3 to 5 together (80% of the total original dataset) will be considered as a train dataset. Similarly, Iterations 3 to 5 work. … (4.16)

- Column 7 represents value of the performance of different iterations. (4.17)

- The value of the average of the performances of Iterations 1 to 5 is represented by Column 8. … (4.18)

Table 4.1: A typical look of a 5-fold cross-validation

	Complete dataset from folds 1 to 5						
	Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**		
Iteration 1	Test set	Train set	Train set	Train set	Train set	Performance for Iteration 1	Average Performance
Iteration 2	Train set	Test set	Train set	Train set	Train set	Performance for Iteration 2	
Iteration 3	Train set	Train set	Test set	Train set	Train set	Performance for Iteration 3	
Iteration 4	Train set	Train set	Train set	Test set	Train set	Performance for Iteration 4	
Iteration 5	Train set	Train set	Train set	Train set	Test set	Performance for Iteration 5	

So, if F1, F2, F3, F4 and F5 denote Folds 1 to 5 then a typical 5-fold cross-validation can be summarised as follows.

Iteration number	**Training Dataset**	**Test Dataset**
Iteration 1	F2 + F3 + F4 + F5	F1
Iteration 2	F1 + F3 + F4 + F5	F2
Iteration 3	F1 + F2 + F4 + F5	F3
Iteration 4	F1 + F2 + F3 + F5	F4
Iteration 5	F1 + F2 + F3 + F4	F5

Let us do one example.

Example 1: **Question based on 5-Folds Cross-Validation**: Suppose you have a dataset with 20 rows and 6 features where each feature is represented by a column. So, you have a data frame having 20 rows and 6 columns. Assume that 20 rows of the data frame are randomly shuffled if you want to apply cross-validation with 5 folds. Specify five folds and summarise the 5 iterations for this data frame.

Solution: Since it is given that 20 rows of the data frame are randomly shuffled. It is also given that we have to make 5 folds, which implies that each fold will have $\frac{20}{5} = 4$ rows. So, we can define 5 folds as follows.

Fold 1 is constituted by the first four rows, so Fold 1 $\rightarrow$ rows 1 to 4

Fold 2 is constituted by the rows 5 to 8, so Fold 2 $\rightarrow$ rows 5 to 8

Fold 3 is constituted by the rows 9 to 12, so Fold 3 $\rightarrow$ rows 9 to 12

Fold 4 is constituted by the rows 13 to 16, so Fold 4 $\rightarrow$ rows 13 to 16

Fold 5 is constituted by the rows 17 to 20, so Fold 5 $\rightarrow$ rows 17 to 20

Five iterations can be summarised as follows.

Iteration number	Training Dataset	Test Dataset
Iteration 1	F2 + F3 + F4 + F5	F1
Iteration 2	F1 + F3 + F4 + F5	F2
Iteration 3	F1 + F2 + F4 + F5	F3
Iteration 4	F1 + F2 + F3 + F5	F4
Iteration 5	F1 + F2 + F3 + F4	F5

Now, you can try the following Self-Assessment Question.

SAQ 1

Question based on 10-Folds Cross-Validation: Suppose you have a dataset with 80 rows and 5 features where each feature is represented by a column. So, you have a data frame having 80 rows and 5 columns. Assume that 80 rows of the data frame are randomly shuffled if you want to apply cross-validation with 10 folds. Specify ten folds and summarise the 10 iterations for this data frame.

4.3 TYPES OF CROSS VALIDATION

In the previous section, you have studied what is cross-validation and why we need it. There are many types of cross-validation which are discussed as follows. Let us first list types of cross-validation.

- k-Fold Cross-Validation … (4.19a)
- Leave-One-Out Cross-Validation (LOOCV) … (4.19b)
- Hold-Out Method … (4.19c)
- Stratified k-Fold Cross-Validation … (4.19d)
- Repeated k-Fold Cross-Validation … (4.19e)
- Time Series Cross-Validation … (4.19f)

Let us explain these taken one at a time as follows.

k-Fold Cross-Validation: You know that in cross-validation, we partition the given dataset into many disjoint subsets of equal size and each subset is known as a fold. If we have k-folds: F1, F2, F3, …, Fk each of equal size, then it is known as k-fold cross-validation. In k-fold cross-validation, we train the model on k – 1 folds and test on the remaining one-fold. This process has k iterations explained as follows:

Iteration 1: Keep fold F1 aside as a test dataset and consider all remaining subsets/folds F2 to Fk together as a training dataset. Obtain performance measure.

Iteration 2: Keep fold F2 aside as a test dataset and consider all remaining subsets/folds F1, F2 to Fk together as a training dataset. Obtain performance measure.

and so on.

Iteration k: Keep fold Fk aside as a test dataset and consider all remaining subsets/folds F1 to F(k – 1) together as a training dataset. Obtain performance measure.

After that we take average of these k performance measures. This average performance measure is better than a single performance measure. … (4.20)

Leave-One-Out Cross-Validation (LOOCV): If each fold has a single observation, then it is known as **leave-one-out cross-validation**. In this case there will be as many folds as the number of rows in the data frame. This cross-validation is useful for small datasets. … (4.21)

For example, if we have n rows in the data frame, then

k = number of folds = number of rows in the data frame = n

Also, number of iterations = number of rows in the data frame = n, where in each iteration we:

- take one row of the data frame as the test set
- take all remaining (n – 1) rows of the data frame for training
- repeat this process n times.

Hold-Out Cross-Validation: If the number of folds is 2, then it is known as hold-out cross-validation. That is here data is split once into a training dataset and a testing dataset. In hold-out cross-validation generally sizes of two folds are not equal. Usually, a 70 : 30 or 75 : 25 or 80 : 20 (train set : test set) split is done. Recall that in Units 2 and 3, we were applying hold-out cross-validation in our discussion. … (4.22)

Stratified k-Fold Cross-Validation: Stratified k-fold cross-validation is preferred when we have imbalanced classification. That is, we have prevalence problem. So, it is the same as k-fold cross-validation but stratified k-fold cross-validation ensures that each fold has the same class proportion as the full dataset. … (4.23)

Repeated k-Fold Cross-Validation: If k-fold cross-validation is repeated multiple times with different random splits then it is known as repeated k-fold cross-validation. … (4.24)

Time Series Cross-Validation: We know that in time series data order of data matters. So, in time series data we do not shuffled rows of data frame randomly. We keep the rows in the order of time. Keeping this in view training of the model is always done on past data and testing is done on future data and training window expand step by step. That is training window progressively increase. Here, we discuss two types of time series cross-validation explained as follows. … (4.25)

Expanding Window Cross-Validation or Rolling Window Cross-Validation: We know that in time series data must stay in time order of data so no random shuffling is done. Keeping this in view training of the model is always done on past data and testing is done on future data and training window expand by one observation in each iteration. Testing observation is always future observation in each iteration. This is explained with the help of an example in Terminal Question 2. … (4.26)

Sliding Window Cross-Validation or Fixed Size Window Cross-Validation: We know that in time series data must stay in time order of data so no random shuffling is done. Keeping this in view training of the model is always done on past data and testing is done on future data. Size of training window remains constant but in successive iteration one oldest observation is removed and one next observation is included. This process continues till we have data to apply this procedure. Like expanding window cross-validation testing observation is always future observation in each iteration. This is explained with the help of an example in Terminal Question 2. … (4.27)

Let us discuss some examples to explain types of cross-validation.

Example 2: **Question based on Leave-One-out Cross-Validation**: Suppose you have 4 rows in your data set given as follows.

Height of Son (in cm)	Height of Father (in cm)
177	180
166	165
172	175
164	160

Assume that 4 rows of the data are randomly shuffled. Explain the procedure of leave-one-out cross-validation for this dataset by considering simple linear regression model as your machine learning algorithm. Specify folds and summaries the iterations for this data frame.

Solution: Since it is given that 4 rows of the data are randomly shuffled. If it is not given to us still the procedure of LOOCV remains the same because in LOOCV each row is considered as a test set. In LOOCV, we know that

k = number of folds = number of rows in the data frame = 4

So, we have to make 4 folds, which implies that each fold will have $4/4=1$ rows. So, we can define 4 folds as follows.

Fold 1 is constituted by the first row, so Fold 1:	177	180
Fold 2 is constituted by the second row, so Fold 2:	166	165
Fold 3 is constituted by the third row, so Fold 3:	172	175
Fold 4 is constituted by the fourth row, so Fold 4:	164	160

Machine learning algorithm to be used here to explain the procedure is simple linear regression. So, our model will look like this

Y = a + bX

Four iterations can be summarised as follows.

Iteration number	**Training Dataset**	**Test Dataset**	**Predicted Value**	**Error used to calculate Performance Measure**	**Average Performance measure (In regression models it is Root Mean Square Error) (RMSE)**
Iteration 1	F2 + F3 + F4	F1	$\hat{y}_1$	$e_1 = y_1 - \hat{y}_1$	RMSE $= \sqrt{\frac{1}{4}\sum_{i=1}^{4} e_i^2}$
Iteration 2	F1 + F3 + F4	F2	$\hat{y}_2$	$e_2 = y_2 - \hat{y}_2$	
Iteration 3	F1 + F2 + F4	F3	$\hat{y}_3$	$e_3 = y_3 - \hat{y}_3$	
Iteration 4	F1 + F2 + F3	F4	$\hat{y}_4$	$e_4 = y_4 - \hat{y}_4$	

Now, you can try the following Self-Assessment Question.

SAQ 2

Question based on Stratified 4-Folds Cross-Validation: Suppose you have a dataset given as follows.

x	15	17	16	13	9	12	18	21	23	20	19	14	25	27	21	22
y	A	A	A	A	A	A	A	A	A	A	A	A	B	B	B	B

In this dataset target variable y has two categories A and B. Note that these categories are not equally proportional in this dataset. Ration between categories A and B is 12 : 4 or 3 : 1. So, you should apply stratified k-fold cross-validation to ensure that each fold has the same proportion as the full dataset. Explain stratified k-fold cross-validation by taking value of k = 4. Also, summaries the iterations for this dataset.

4.4 CROSS-VALIDATION IN R

In previous section, you have seen that cross-validation gives an opportunity to each row of the data frame to be part of test dataset in some iteration. The cost that we have to pay for it is that it increases time of training and testing the machine learning algorithm. But we are lucky that we have not to do it manually, we are surviving in the era of softwares. Specifically, R software which makes our life easy in doing all this. We will use createFolds(), trainControl() and train() functions of caret package to do cross-validation. In this section, you will learn to apply these functions in R. … (4.28)

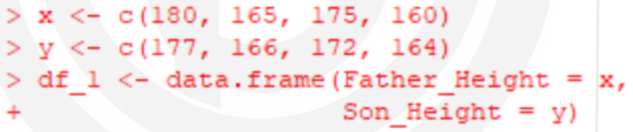
Let us do one example.

Example 3: **Question based Leave-One-Out Cross-Validation**: Suppose you have 4 rows in your data set given as follows.

Height of Son (in cm)	Height of Father (in cm)
177	180
166	165
172	175
164	160

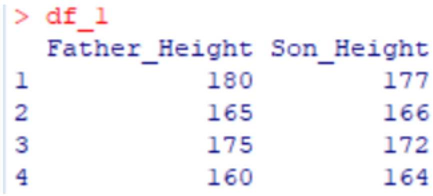
Assume that 4 rows of the data are randomly shuffled. Explain the procedure of leave-one-out cross-validation for this dataset by considering simple linear regression model as your machine learning algorithm. Obtain the average RMSE using R.

Solution: First of all, let us create a data frame in R having above data in two columns. This can be done as follows.



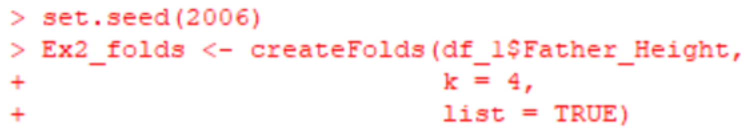
```
> x <- c(180, 165, 175, 160)
> y <- c(177, 166, 172, 164)
> df_1 <- data.frame(Father_Height = x,
+                    Son_Height = y)
```

Let us have a look at our data frame df_1. This can be seen by simply typing df_1 on R console and hitting enter key. Screenshot of the output is shown as follows.



```
> df_1
  Father_Height Son_Height
1           180        177
2           165        166
3           175        172
4           160        164
```

Now, let us create folds. This can be seen as follows.

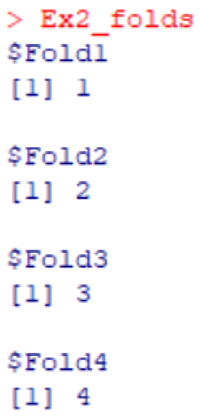


```
> set.seed(2006)
> Ex2_folds <- createFolds(df_1$Father_Height,
+                          k = 4,
+                          list = TRUE)
```

… (4.29)

Let us see which rows are in each fold. This can be seen by simply typing Ex2_folds on R console and hitting enter key. Screenshot of the output is shown as follows.

ignou
THE PEOPLE'S
UNIVERSITY

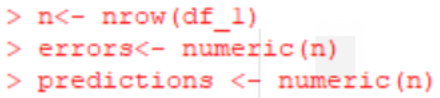


```
> Ex2_folds
$Fold1
[1] 1

$Fold2
[1] 2

$Fold3
[1] 3

$Fold4
[1] 4
```

… (4.30)

We see that Folds: 1, 2, 3 and 4 have rows 1, 2, 3 and 4 respectively.

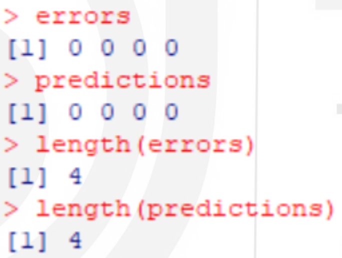
Next, let us create errors and predictions vectors each of length equal to the number of folds which is equal to number of rows in the data frame df_1 in this example. This can be done as follows.



```
> n<- nrow(df_1)
> errors<- numeric(n)
> predictions <- numeric(n)
```

… (4.31)

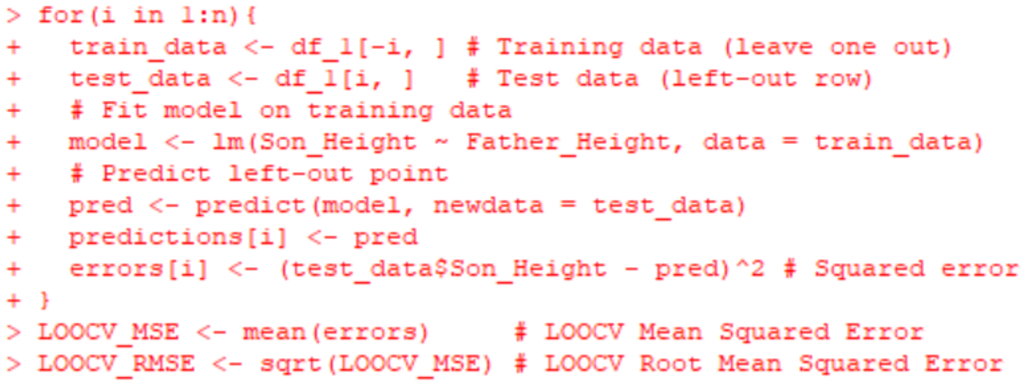
We have created errors and predictions vectors each of length 4, you can check by running the following on R console.



```
> errors
[1] 0 0 0 0
> predictions
[1] 0 0 0 0
> length(errors)
[1] 4
> length(predictions)
[1] 4
```

… (4.32)

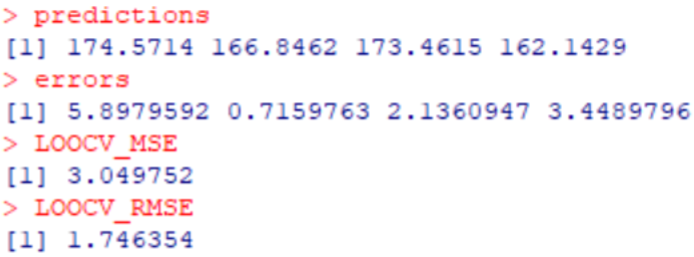
Now, we can apply for loop to obtain (i) errors vector (ii) predictions vector (iii) mean square error (LOOCV_MSE) and (iv) root mean square error (LOOCV_RMSE) of the lm() machine learning algorithm.



```
> for(i in 1:n){
+   train_data <- df_1[-i, ] # Training data (leave one out)
+   test_data <- df_1[i, ]   # Test data (left-out row)
+   # Fit model on training data
+   model <- lm(Son_Height ~ Father_Height, data = train_data)
+   # Predict left-out point
+   pred <- predict(model, newdata = test_data)
+   predictions[i] <- pred
+   errors[i] <- (test_data$Son_Height - pred)^2 # Squared error
+ }
> LOOCV_MSE <- mean(errors)      # LOOCV Mean Squared Error
> LOOCV_RMSE <- sqrt(LOOCV_MSE) # LOOCV Root Mean Squared Error
```

… (4.33)

These codes have created all the required information. Required information can be obtained by simply typing errors, predictions, errors, LOOCV_MSE and LOOCV_RMSE on R console and hitting the enter key. Screenshot of the output is shown as follows.



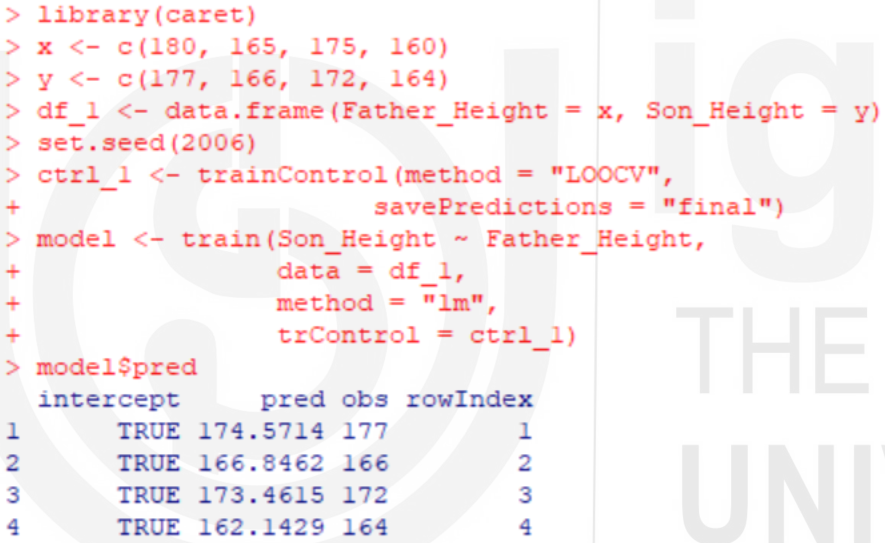
```
> predictions
[1] 174.5714 166.8462 173.4615 162.1429
> errors
[1] 5.8979592 0.7159763 2.1360947 3.4489796
> LOOCV_MSE
[1] 3.049752
> LOOCV_RMSE
[1] 1.746354
```

… (4.34)

Note that when Fold 1 is considered as a test set then we get maximum error 5.8979592 while if Fold 2 is considered as a test set then we get minimum error 0.7159763. Mean square error gives us the average value of these four errors obtained in four different folds. Root mean square error gives us the square root of mean square error. The vector predictions gives predicted values of the height of son by the simple linear regression model. … (4.35)

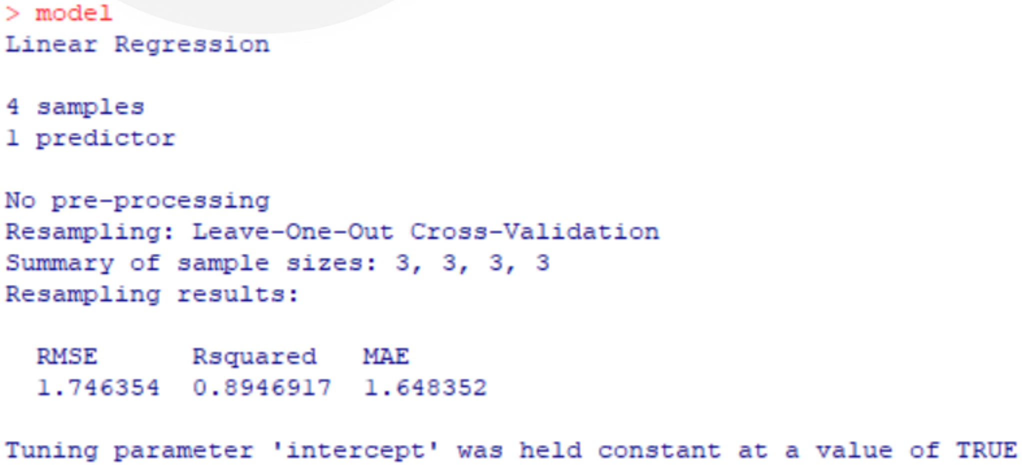
Instead of applying for loop, we can also obtain the same information by using built-in function trainControl() and train() of caret package. This is done as follows. … (4.36)

Alternative Method:



```
> library(caret)
> x <- c(180, 165, 175, 160)
> y <- c(177, 166, 172, 164)
> df_1 <- data.frame(Father_Height = x, Son_Height = y)
> set.seed(2006)
> ctrl_1 <- trainControl(method = "LOOCV",
+                        savePredictions = "final")
> model <- train(Son_Height ~ Father_Height,
+                data = df_1,
+                method = "lm",
+                trControl = ctrl_1)
> model$pred
  intercept     pred obs rowIndex
1      TRUE 174.5714 177        1
2      TRUE 166.8462 166        2
3      TRUE 173.4615 172        3
4      TRUE 162.1429 164        4
```

… (4.37)



```
> model
Linear Regression

4 samples
1 predictor

No pre-processing
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 3, 3, 3, 3
Resampling results:

  RMSE      Rsquared   MAE
  1.746354  0.8946917  1.648352

Tuning parameter 'intercept' was held constant at a value of TRUE
```

… (4.38)

Interpretation of the output of the model:

Root Mean Square Error (RMSE) and **Mean Absolute Error (MAE)** both give average distance of error from 0 but their procedure of removing negative sign of error differ. In RMSE, we square the difference of respective actual values and predicted values of the target variable to remove negative sign, then take average and finally take square root. In MAE, we take absolute value of the difference of respective actual values and predicted values of the target variable to remove negative sign, then take average. In mathematical notation, MAE can be defined as follows. … (4.39)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \qquad \text{… (4.40)}$$

, where

$y_i =$ actual value

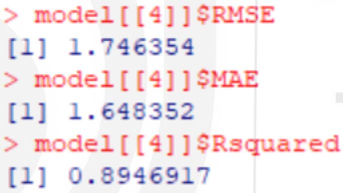
$\hat{y}_i =$ predicted value

$\left|y_i - \hat{y}_i\right| =$ absolute error for each point

$n =$ total number of observations

R-Squared: You are familier with the interpretation of R-squared from the course MST-017. Here, its value is 0.8946917 which means feature Father_Height is able to explain 89.47% of the variation in the target variable Son_Height. … (4.41)

After running the model like (4.37), you can also obtain values of RMSE, Rsquared and MAE as follows.



```
> model[[4]]$RMSE
[1] 1.746354
> model[[4]]$MAE
[1] 1.648352
> model[[4]]$Rsquared
[1] 0.8946917
```

… (4.42)

Note that we obtain the same information by using trainControl() and train() functions of caret package in R as we obtained using for loop. But in future we will apply functions of caret package in doing such calculation. … (4.43)

Now, you can try the following Self-Assessment Question.

SAQ 3

Question based on 5-Folds Cross-Validation: Consider the built-in mtcars dataset in R. Using mpg as the target variable and all remaining variables as predictors/features, apply the linear regression machine learning algorithm with 5-fold cross-validation. Show the position of the rows selected in each of the five folds. Obtain the complete set of predicted values and compute the RMSE, MAE and R-squared of the model.

4.5 SUMMARY

A brief summary of what we have covered in this unit is given as follows:

In Units 2 and 3 we partitioned the given dataset into two parts, known as the training dataset and the testing or test dataset. Our main objectives of doing that were to:

- **Train the Model on Known Data**: Training dataset is used for learning patterns, rules, and relationship from data as well as estimating parameters, and finally building the predictive model.

- **Evaluate Model Performance on Unseen Data**: The Model is trained using the training dataset, so the testing dataset is a new or unseen dataset for the learned model. So, the test dataset is used to check how well the trained model performs on unseen data. The ultimate goal of a machine learning model/algorithm is to make prediction on unseen data. So, using the test dataset, we can estimate the accuracy and predictive ability of the model. In short, we can say that the testing dataset acts like future real-world data to estimate how the trained model will behave. That is, the testing dataset can be treated as a reliable estimate of how the model will behave when deployed on real, future data.

- **To Detect Overfitting and Underfitting**: In Units 2 and 3 you have learnt to obtain many metrics to test the performance of the model. On the basis of these metrics you can identify whether the model is underfitting or overfitting explained as follows.
  - **Overfitting**: We say that a model is overfitting if it performs well on the training dataset but poorly on testing dataset.
  - **Underfitting**: We say that a model is underfitting if it performs poorly on both training and testing datasets.

- **To Compare or Select the Best Model**: In Unit 3, you learnt to obtain the area under the ROC curve (AUC). By calculating the AUC for the test dataset for all models under comparison, we can say which model performs well on the test dataset compared to the other models.

- **Cross-Validation**: In cross-validation, instead of partitioning the given dataset into a single training dataset and a single testing dataset, we partition the given dataset into many disjoint subsets of equal size (as far as possible) and each subset is known as a fold. After that we proceed iteratively as follows.
  - **Iteration 1**: We keep first subset/fold aside as a test dataset and consider all remaining subsets/folds (Folds 2 to last fold) as a training dataset. Obtain performance measure.
  - **Iteration 2**: We keep second subset/fold aside as a test dataset and consider all remaining subsets/folds (Fold 1, 3, 4 to last fold) as a training dataset. Obtain performance measure.
  - and so on.
  - **Last Iteration**: We keep last subset/fold aside as a test dataset and consider all remaining subsets/folds (Fold 1 to second last fold) as a training dataset. Obtain performance measure.

  So, in this process we will get as many estimates of the performance measure on different test sets as the number of subsets/folds. After training and testing the model as many times as the number of subsets/folds, we take average of the performance measure. This process is known as cross-validation.

- **Types of Cross-Validation**: There are many types of cross-validation which are defined as follows.
  - **k-Fold Cross-Validation**: You know that in cross-validation, we partition the given dataset into many disjoint subsets of equal size and each subset is known as a fold. If we have k-folds: F1, F2, F3, …, Fk each of equal size, then it is known as k-fold cross-validation. In k-fold cross-validation, we train the model on k – 1 folds and test on the remaining one-fold. This process has k iterations explained as follows:
    - **Iteration 1**: Keep fold F1 aside as a test dataset and consider all remaining subsets/folds F2 to Fk together as a training dataset. Obtain performance measure.
    - **Iteration 2**: Keep fold F2 aside as a test dataset and consider all remaining subsets/folds F1, F2 to Fk together as a training dataset. Obtain performance measure.
    - and so on.
    - **Iteration k**: Keep fold Fk aside as a test dataset and consider all remaining subsets/folds F1 to F(k – 1) together as a training dataset. Obtain performance measure.
    - After that we take average of these k performance measures. This average performance measure is better than a single performance measure.
  - **Leave-One-Out Cross-Validation (LOOCV)**: If each fold has a single observation, then it is known as **leave-one-out cross-validation**. In this case there will be as many folds as the number of rows in the data frame. This cross-validation is useful for small datasets.
  - **Hold-Out Cross-Validation**: If the number of folds is 2, then it is known as hold-out cross-validation. That is here data is split once into a training dataset and a testing dataset. In hold-out cross-validation generally sizes of two folds are not equal. Usually, a 70 : 30 or 75 : 25 or 80 : 20 (train set : test set) split is done. Recall that in Units 2 and 3, we were applying hold-out cross-validation in our discussion.
  - **Stratified k-Fold Cross-Validation**: Stratified k-fold cross-validation is preferred when we have imbalanced classification. That is, we have prevalence problem. So, it is the same as k-fold cross-validation but stratified k-fold cross-validation ensures that each fold has the same class proportion as the full dataset.
  - **Repeated k-Fold Cross-Validation**: If k-fold cross-validation is repeated multiple times with different random splits then it is known as repeated k-fold cross-validation.

- **Time Series Cross-Validation**: We know that in time series data order of data matters. So, in time series data we do not shuffled rows of data frame randomly. We keep the rows in the order of time. Keeping this in view training of the model is always done on past data and testing is done on future data and training window expand step by step. That is training window progressively increase. Time series cross-validation can further be classified into two types explained as follows.

    - **Expanding Window Cross-Validation or Rolling Window Cross-Validation**: We know that in time series data must stay in time order of data so no random shuffling is done. Keeping this in view training of the model is always done on past data and testing is done on future data and training window expand by one observation in each iteration. Testing observation is always future observation in each iteration. This is explained with the help of an example in Terminal Question 2.

    - **Sliding Window Cross-Validation or Fixed Size Window Cross-Validation**: We know that in time series data must stay in time order of data so no random shuffling is done. Keeping this in view training of the model is always done on past data and testing is done on future data. Size of training window remains constant but in successive iteration one oldest observation is removed and one next observation is included. This process continues till we have data to apply this procedure. Like expanding window cross-validation testing observation is always future observation in each iteration. This is explained with the help of an example in Terminal Question 2.

- Cross-validation gives an opportunity to each row of the data frame to be part of test dataset in some iteration. The cost that we have to pay for it is that it increases time of training and testing the machine learning algorithm. But we are lucky that we have not to do it manually, we are surviving in the era of softwares. Specifically, R software which makes our life easy in doing all this. We have learnt to apply createFolds(), trainControl() and train() functions of caret package to do cross-validation in R.

4.6 TERMINAL QUESTIONS

1. **Question based on 10-Folds Cross-Validation**: Suppose you have a dataset with 120 rows and 15 features where each feature is represented by a column. So, you have a data frame having 120 rows and 15 columns. If you want to apply cross-validation with 10 folds. Specify 10 folds and summarise the 10 iterations for this data frame.

2. **Question based on Time Series Cross-Validation**: Explain different types of time series cross-validation with the help of the following example where profit in crores in the first ten years of a company is mentioned as follows.

Time	1	2	3	4	5	6	7	8	9	10
Profit (in crore)	15	17	18	21	25	26	29	33	36	40

3. **Question based on 5-Folds Cross-Validation**: Consider the built-in mtcars dataset in R. Using mpg as the target variable and only wt and hp variables as predictors/features, apply the linear regression machine learning algorithm with 5-fold cross-validation. Obtain the complete set of predicted values and compute the RMSE of the model.

4. **Question based on 5-Folds Cross-Validation**: Consider the built-in mtcars dataset in R. Using mpg as the target variable and only wt and hp variables as predictors/features, apply the linear regression machine learning algorithm with 5-fold cross-validation. You have already obtained RMSE, MAE and Rsquared for model in Terminal Question 3. Now, obtain RMSE and MAE for each fold.

5. **Question based on Repeated 5-Folds Cross-Validation**: Consider the built-in mtcars dataset in R. Using mpg as the target variable and only wt and hp variables as predictors/features, apply the linear regression machine learning algorithm with repeated 5-fold cross-validation. Take 3 as the value of repetition. Obtain RMSE, MAE and Rsquared for the model. Also, obtain RMSE, MAE for each fold and each repeat.

6. Consider the built-in AirPassengers dataset in R. Begin by displaying the complete dataset. Convert it into a data frame named u4_TQ6_df with two variables: y, representing the number of passengers, and t, representing the time index from 1 to 144 (where 1 corresponds to January 1949, 2 to February 1949, 3 to March 1949, …, and 144 to December 1960). Using y as the target variable and the time variable t as predictor/feature, apply a linear regression machine learning algorithm with rolling window cross-validation by setting the initial training window (initialWindow) to 60 months and the testing window (horizon) to 12 months. Obtain the RMSE and MAE of the model.

4.7 SOLUTIONS/ANSWERS

Self-Assessment Questions (SAQs)

1. Here, it is not given that 80 rows of the data frame are randomly shuled. So, first, we have to randomly shuffled the 80 rows. Since it is given that 80 rows of the data frame are randomly shuffled. It is also given that we have to make 10 folds, which implies that each fold will have $80/10 = 8$ rows. So, we can define 10 folds as follows.

Fold 1 is constituted by the first eight rows, so Fold 1 $\rightarrow$ rows 1 to 8

Fold 2 is constituted by the rows 9 to 16, so Fold 2 $\rightarrow$ rows 9 to 16

Fold 3 is constituted by the rows 17 to 24, so Fold 3 $\rightarrow$ rows 17 to 24

Fold 4 is constituted by the rows 25 to 32, so Fold 4 → rows 25 to 32

Fold 5 is constituted by the rows 33 to 40, so Fold 5 → rows 33 to 40

Fold 6 is constituted by the rows 41 to 48, so Fold 6 → rows 41 to 48

Fold 7 is constituted by the rows 49 to 56, so Fold 7 → rows 49 to 56

Fold 8 is constituted by the rows 57 to 64, so Fold 8 → rows 57 to 64

Fold 9 is constituted by the rows 65 to 72, so Fold 9 → rows 65 to 72

Fold 10 is constituted by the rows 73 to 80, so Fold 10 → rows 73 to 80

Ten iterations can be summarised as follows.

Iteration number	**Training Dataset**	**Test Dataset**
Iteration 1	F2 + F3 + F4 + F5 + F6 + F7 + F8 + F9 + F 10	F1
Iteration 2	F1 + F3 + F4 + F5 + F6 + F7 + F8 + F9 + F 10	F2
Iteration 3	F1 + F2 + F4 + F5 + F6 + F7 + F8 + F9 + F 10	F3
Iteration 4	F1 + F2 + F3 + F5 + F6 + F7 + F8 + F9 + F 10	F4
Iteration 5	F1 + F2 + F3 + F4 + F6 + F7 + F8 + F9 + F 10	F5
Iteration 6	F1 + F2 + F3 + F4 + F5 + F7 + F8 + F9 + F 10	F6
Iteration 7	F1 + F2 + F3 + F4 + F5 + F6 + F8 + F9 + F 10	F7
Iteration 8	F1 + F2 + F3 + F4 + F5 + F6 + F7 + F9 + F 10	F8
Iteration 9	F1 + F2 + F3 + F4 + F5 + F6 + F7 + F8 + F 10	F9
Iteration 10	F1 + F2 + F3 + F4 + F5 + F6 + F7 + F8 + F9	F10

2. Generally, variables in a dataset are in columns and each row represents data of a subject. But in the present dataset variables are represented by rows and data of 16 subjects are represented by columns 2 to 17. Here, variable y is a categorical variable having two categories A and B. But categories A and B in this dataset are not equally proportional. So, we are asked to perform stratified k-fold cross-validation with k = 4.

   First of all, we assume that 12 columns of category A from Column 2 to 13 and 4 columns of category B from Column 14 to 17 are arranged randomly. The total number of observations is 16 and we have to form 4 folds so each fold will have 16/4 = 4 observations. But we know that stratified k-fold cross-validation ensures that each fold has the same class proportion as the full dataset. Here, full dataset has 12 observations of category A and 4 observations of category B. So,

   Number of observations of category A to the number of observations of category B is 12 : 4 or 3 : 1. So, in our 4 folds we have to maintained this ratio between categories A and B in each fold. It means each fold will contain 3 observations of category A and 1 observation of category B.

So, we can define 4 folds as follows.

Fold 1 is constituted by observations present in columns 2 to 4 and column 14, so Fold 1:	15	17	16	25
	A	A	A	B
Fold 2 is constituted by observations present in columns 5 to 7 and column 15, so Fold 2:	13	9	12	27
	A	A	A	B
Fold 3 is constituted by observations present in columns 8 to 10 and column 16, so Fold 3:	18	21	23	21
	A	A	A	B
Fold 4 is constituted by observations present in columns 11 to 13 and column 17, so Fold 4:	20	19	14	22
	A	A	A	B

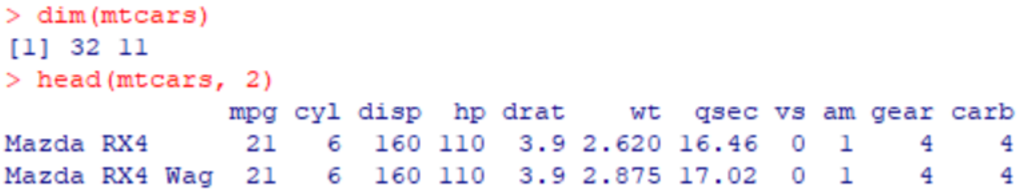
Note that in each of the four folds ratio between the number of observations of categories A and B is 3 : 1 as required due to definition of stratified 4-fold cross-validation.

Four iterations can be summarised as follows.

Iteration number	Training Dataset	Test Dataset
Iteration 1	F2 + F3 + F4	F1
Iteration 2	F1 + F3 + F4	F2
Iteration 3	F1 + F2 + F4	F3
Iteration 4	F1 + F2 + F3	F4

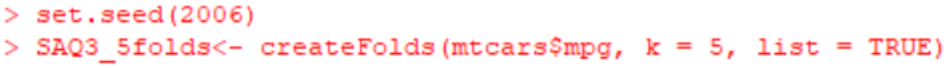
ignou
THE PEOPLE'S
UNIVERSITY

3. By this time, you have become familiar with mtcars dataset because we have used this dataset many times. Still let run dim() function which tells number of rows and columns in this dataset. We know the answer of this question will be 32 and 11. Next we run the function head() to see first two rows of this dataset. By default head() function shows first six rows. So, to see only first two rows, we have to specify second argument equal to 2. Screenshot of the output is shown as follows.



```
> dim(mtcars)
[1] 32 11
> head(mtcars, 2)
              mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4      21   6  160 110  3.9 2.620 16.46  0  1    4    4
Mazda RX4 Wag  21   6  160 110  3.9 2.875 17.02  0  1    4    4
```

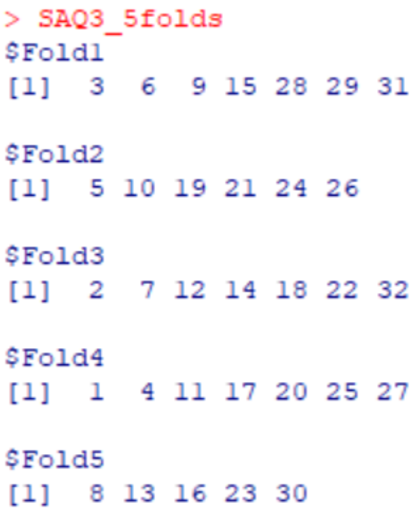
Now, let us create 5 folds. This can be done as follows.



```
> set.seed(2006)
> SAQ3_5folds<- createFolds(mtcars$mpg, k = 5, list = TRUE)
```

… (4.44)

Let us see which rows are in each fold. This can be seen by simply typing SAQ_3_5_folds on R console and hitting enter key. Screenshot of the output is shown as follows.

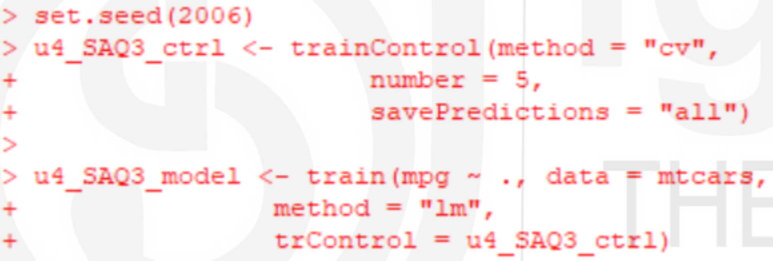


```
> SAQ3_5folds
$Fold1
[1]  3  6  9 15 28 29 31

$Fold2
[1]  5 10 19 21 24 26

$Fold3
[1]  2  7 12 14 18 22 32

$Fold4
[1]  1  4 11 17 20 25 27

$Fold5
[1]  8 13 16 23 30
```

… (4.45)

We see that Fold 1 contains seven rows with positions 3, 6, 9, 15, 28, 29 and 31. Fold 2 contains six rows with positions 5, 10, 19, 21, 24 and 26. Similarly, you can see positions of rows of the Folds 3 to 5 in (4.45).

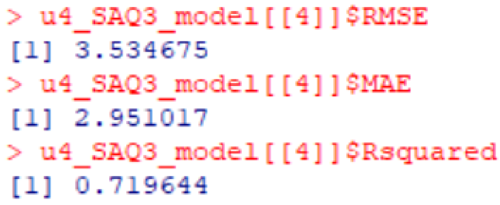
As promised in (4.43), we will use trainControl() and train() functions of caret package in R to train lm() machine learning algorithm on mtcars dataset. This can be done as follows.



```
> set.seed(2006)
> u4_SAQ3_ctrl <- trainControl(method = "cv",
+                              number = 5,
+                              savePredictions = "all")
>
> u4_SAQ3_model <- train(mpg ~ ., data = mtcars,
+                        method = "lm",
+                        trControl = u4_SAQ3_ctrl)
```

… (4.46)

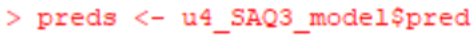
These codes have created all the required information. Root mean square error (RMSE), mean absolute error (MAE) and R-squared can be obtained by simply typing u_4_SAQ_3_model[[4]]$RMSE, u_4_SAQ_3_model[[4]]$MAE and u_4_SAQ_3_model[[4]]$Rsquared on R console and hitting the enter key. Screenshot of the output is shown as follows.



```
> u4_SAQ3_model[[4]]$RMSE
[1] 3.534675
> u4_SAQ3_model[[4]]$MAE
[1] 2.951017
> u4_SAQ3_model[[4]]$Rsquared
[1] 0.719644
```

… (4.47)

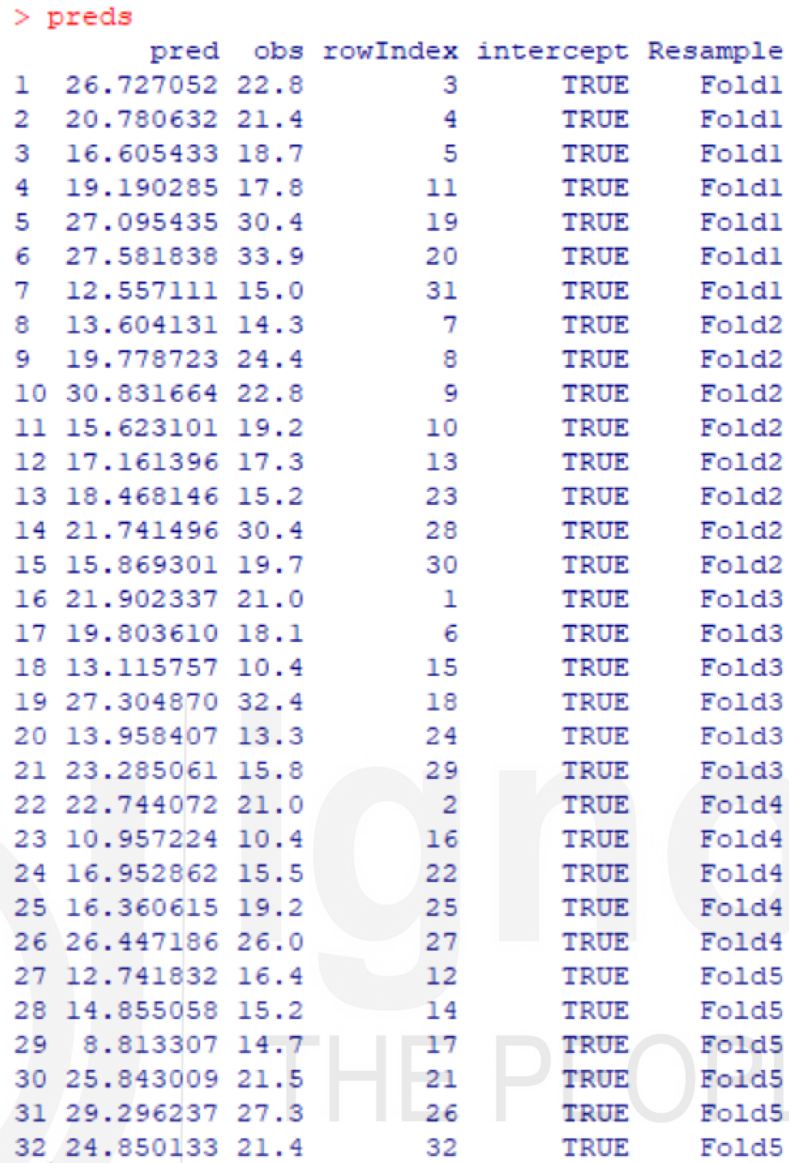
Complete set of predicted values can be stored in the variable preds as follows.



```
> preds <- u4_SAQ3_model$pred
```

… (4.48)

Now, all predicted values can be seen by typing preds on R console and hitting enter key. The screenshot of the output is shown as follows.



```
> preds
        pred  obs rowIndex intercept Resample
1  26.727052 22.8        3      TRUE    Fold1
2  20.780632 21.4        4      TRUE    Fold1
3  16.605433 18.7        5      TRUE    Fold1
4  19.190285 17.8       11      TRUE    Fold1
5  27.095435 30.4       19      TRUE    Fold1
6  27.581838 33.9       20      TRUE    Fold1
7  12.557111 15.0       31      TRUE    Fold1
8  13.604131 14.3        7      TRUE    Fold2
9  19.778723 24.4        8      TRUE    Fold2
10 30.831664 22.8        9      TRUE    Fold2
11 15.623101 19.2       10      TRUE    Fold2
12 17.161396 17.3       13      TRUE    Fold2
13 18.468146 15.2       23      TRUE    Fold2
14 21.741496 30.4       28      TRUE    Fold2
15 15.869301 19.7       30      TRUE    Fold2
16 21.902337 21.0        1      TRUE    Fold3
17 19.803610 18.1        6      TRUE    Fold3
18 13.115757 10.4       15      TRUE    Fold3
19 27.304870 32.4       18      TRUE    Fold3
20 13.958407 13.3       24      TRUE    Fold3
21 23.285061 15.8       29      TRUE    Fold3
22 22.744072 21.0        2      TRUE    Fold4
23 10.957224 10.4       16      TRUE    Fold4
24 16.952862 15.5       22      TRUE    Fold4
25 16.360615 19.2       25      TRUE    Fold4
26 26.447186 26.0       27      TRUE    Fold4
27 12.741832 16.4       12      TRUE    Fold5
28 14.855058 15.2       14      TRUE    Fold5
29  8.813307 14.7       17      TRUE    Fold5
30 25.843009 21.5       21      TRUE    Fold5
31 29.296237 27.3       26      TRUE    Fold5
32 24.850133 21.4       32      TRUE    Fold5
```

… (4.49)

Interpretation of the output of the model:

Root Mean Square Error (RMSE) and **Mean Absolute Error (MAE)** both give average distance of error from 0 but their procedure of removing negative sign of error differ. In RMSE, we square the difference of respective actual values and predicted values of the target variable to remove negative sign, then take average and finally take square root. In MAE, we take absolute value of the difference of respective actual values and predicted values of the target variable to remove negative sign, then take average. … (4.50)

R-Squared: You are familier with the interpretation of R-squared from the course MST-017. Here, its value is 0.719644 which means all features of the dataset mtcars are able to explain 71.96% of the variation in the target variable mpg. … (4.51)

Terminal Questions

1. Here, it is not given that 120 rows of the data frame are randomly shuffled. So, first, we have to randomly shuffle the 120 rows. Now, after doing that, we can follow the same procedure as discussed in Example 1

or SAQ 1. It is also given that we have to make 10 folds, which implies that each fold will have $\frac{120}{10} = 12$ rows. So, we can define 10 folds as follows.

Fold 1 is constituted by the first twelve rows, so Fold 1 → rows 1 to 12

Fold 2 is constituted by the rows 13 to 24, so Fold 2 → rows 13 to 24

Fold 3 is constituted by the rows 25 to 36, so Fold 3 → rows 25 to 36

Fold 4 is constituted by the rows 37 to 48, so Fold 4 → rows 37to 48

Fold 5 is constituted by the rows 49 to 60, so Fold 5 → rows 49 to 60

Fold 6 is constituted by the rows 61 to 72, so Fold 6 → rows 61 to 72

Fold 7 is constituted by the rows 73 to 84, so Fold 7 → rows 73 to 84

Fold 8 is constituted by the rows 85 to 96, so Fold 8 → rows 85 to 96

Fold 9 is constituted by the rows 97 to 108, so Fold 9 → rows 97 to 108

Fold 10 is constituted by the rows 109 to 120, so Fold 10 → rows 109 to 120

Ten iterations can be summarised as follows.

Iteration number	**Training Dataset**	**Test Dataset**
Iteration 1	F2 + F3 + F4 + F5 + F6 + F7 + F8 + F9 + F 10	F1
Iteration 2	F1 + F3 + F4 + F5 + F6 + F7 + F8 + F9 + F 10	F2
Iteration 3	F1 + F2 + F4 + F5 + F6 + F7 + F8 + F9 + F 10	F3
Iteration 4	F1 + F2 + F3 + F5 + F6 + F7 + F8 + F9 + F 10	F4
Iteration 5	F1 + F2 + F3 + F4 + F6 + F7 + F8 + F9 + F 10	F5
Iteration 6	F1 + F2 + F3 + F4 + F5 + F7 + F8 + F9 + F 10	F6
Iteration 7	F1 + F2 + F3 + F4 + F5 + F6 + F8 + F9 + F 10	F7
Iteration 8	F1 + F2 + F3 + F4 + F5 + F6 + F7 + F9 + F 10	F8
Iteration 9	F1 + F2 + F3 + F4 + F5 + F6 + F7 + F8 + F 10	F9
Iteration 10	F1 + F2 + F3 + F4 + F5 + F6 + F7 + F8 + F9	F10

2. We have already explained that time series cross-validation can be classified in two types.

- Expanding Window Cross-Validation or Rolling Window Cross-Validation
- Sliding Window Cross-Validation or Fixed Size Window Cross Validation

Let us explain them one by one as follows.

- **Expanding Window Cross-Validation or Rolling Window Cross-Validation**

Fold	Training Dataset	Testing Dataset
1	First observation 15	Second observation 17
2	First two observations 15, 17	Third observation 18
3	First three observations 15, 17, 18	Fourth observation 21
4	First two observations 15, 17, 18, 21	Fifth observation 25
5	First two observations 15, 17, 18, 21, 25	Sixth observation 26
6	First two observations 15, 17, 18, 21, 25, 26	Seventh observation 29
7	First two observations 15, 17, 18, 21, 25, 26, 29	Eighth observation 33
8	First two observations 15, 17, 18, 21, 25, 26, 29, 33	Ninth observation 36
9	First two observations 15, 17, 18, 21, 25, 26, 29, 33, 36	Tenth observation 40

- **Sliding Window Cross-Validation or Fixed Window Cross-Validation**

  We know that in sliding window cross-validation, training window size is constant. So, to explain the procedure let us consider 5 as the size of sliding window.

Fold	Training Dataset	Testing Dataset
1	First five observations 15, 17, 18, 21, 25	Sixth observation 26
2	Second to sixth observations 17, 18, 21, 25, 26	Seventh observation 29
3	Third to seventh observations 18, 21, 25, 26, 29	Fourth observation 33
4	Fourth to eighth observations 21, 25, 26, 29, 33	Fifth observation 36
5	Fifth to ninth observations 25, 26, 29, 33, 36	Sixth observation 40

Here, it is not given that 120 rows of the data frame are randomly shuffled. So, first, we have to randomly shuffle the 120 rows. Now, after doing that, we can follow the same procedure as discussed in Example 1 or SAQ 1. It is also given that we have to make 10 folds, which implies that each fold will have $\frac{120}{10} = 12$ rows. So, we can define 10 folds as follows.

Fold 1 is constituted by the first twelve rows, so Fold 1 $\rightarrow$ rows 1 to 12

Fold 2 is constituted by the rows 13 to 24, so Fold 2 $\rightarrow$ rows 13 to 24

Fold 3 is constituted by the rows 25 to 36, so Fold 3 $\rightarrow$ rows 25 to 36

Fold 4 is constituted by the rows 37 to 48, so Fold 4 $\rightarrow$ rows 37to 48

Fold 5 is constituted by the rows 49 to 60, so Fold 5 $\rightarrow$ rows 49 to 60

Fold 6 is constituted by the rows 61 to 72, so Fold 6 $\rightarrow$ rows 61 to 72

Fold 7 is constituted by the rows 73 to 84, so Fold 7 $\rightarrow$ rows 73 to 84

Fold 8 is constituted by the rows 85 to 96, so Fold 8 $\rightarrow$ rows 85 to 96

Fold 9 is constituted by the rows 97 to 108, so Fold 9 $\rightarrow$ rows 97 to 108

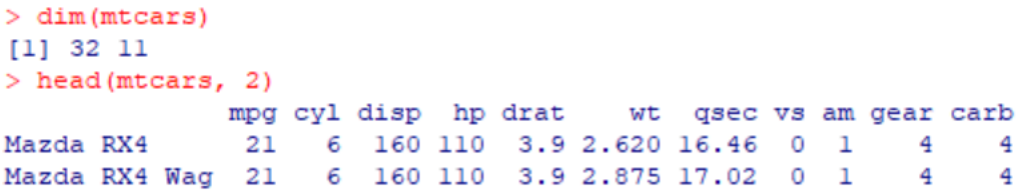
Fold 10 is constituted by the rows 109 to 120, so Fold 10 $\rightarrow$ rows 109 to 120

Ten iterations can be summarised as follows.

Iteration number	**Training Dataset**	**Test Dataset**
Iteration 1	F2 + F3 + F4 + F5 + F6 + F7 + F8 + F9 + F 10	F1
Iteration 2	F1 + F3 + F4 + F5 + F6 + F7 + F8 + F9 + F 10	F2
Iteration 3	F1 + F2 + F4 + F5 + F6 + F7 + F8 + F9 + F 10	F3
Iteration 4	F1 + F2 + F3 + F5 + F6 + F7 + F8 + F9 + F 10	F4
Iteration 5	F1 + F2 + F3 + F4 + F6 + F7 + F8 + F9 + F 10	F5
Iteration 6	F1 + F2 + F3 + F4 + F5 + F7 + F8 + F9 + F 10	F6
Iteration 7	F1 + F2 + F3 + F4 + F5 + F6 + F8 + F9 + F 10	F7
Iteration 8	F1 + F2 + F3 + F4 + F5 + F6 + F7 + F9 + F 10	F8
Iteration 9	F1 + F2 + F3 + F4 + F5 + F6 + F7 + F8 + F 10	F9
Iteration 10	F1 + F2 + F3 + F4 + F5 + F6 + F7 + F8 + F9	F10

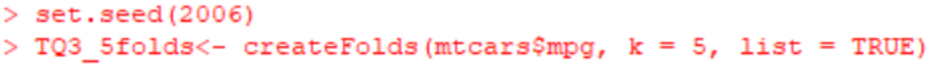
3. By this time, you have become familiar with mtcars dataset because we have used this dataset many times. Still let run dim() function which tells number of rows and columns in this dataset. We know the answer of this question will be 32 and 11. Next we run the function head() to see first two rows of this dataset. By default head() function shows first six rows.

So, to see only first two rows, we have to specify second argument equal to 2. Screenshot of the output is shown as follows.



```
> dim(mtcars)
[1] 32 11
> head(mtcars, 2)
              mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4      21   6  160 110  3.9 2.620 16.46  0  1    4    4
Mazda RX4 Wag  21   6  160 110  3.9 2.875 17.02  0  1    4    4
```

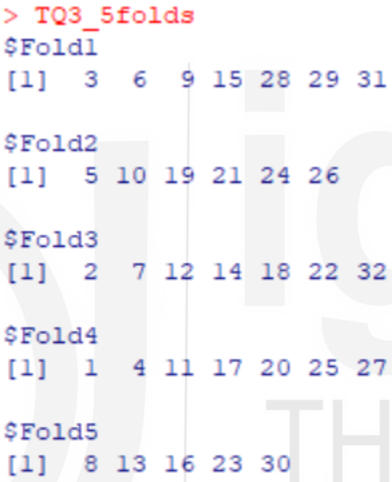
Now, let us create 5 folds. This can be done as follows.



```
> set.seed(2006)
> TQ3_5folds<- createFolds(mtcars$mpg, k = 5, list = TRUE)
```

… (4.52)

Let us see which rows are in each fold. This can be seen by simply typing TQ3_5folds on R console and hitting enter key. Screenshot of the output is shown as follows.

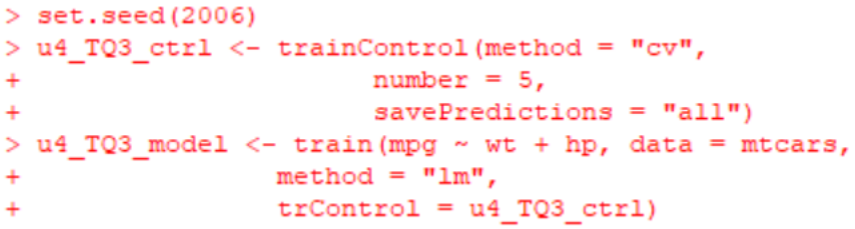


```
> TQ3_5folds
$Fold1
[1]  3  6  9 15 28 29 31

$Fold2
[1]  5 10 19 21 24 26

$Fold3
[1]  2  7 12 14 18 22 32

$Fold4
[1]  1  4 11 17 20 25 27

$Fold5
[1]  8 13 16 23 30
```

… (4.53)

We see that Fold 1 contains seven rows with positions 3, 6, 9, 15, 28, 29 and 31. Fold 2 contains six rows with positions 5, 10, 19, 21, 24 and 26. Similarly, you can see positions of rows of the Folds 3 to 5 in (4.53).

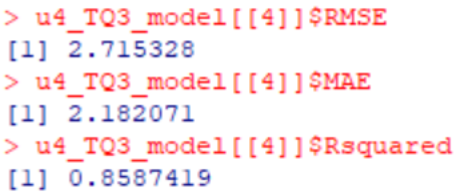
As promised in (4.43), we will use trainControl() and train() functions of caret package in R to train lm() machine learning algorithm on mtcars dataset. This can be done as follows.



```
> set.seed(2006)
> u4_TQ3_ctrl <- trainControl(method = "cv",
+                             number = 5,
+                             savePredictions = "all")
> u4_TQ3_model <- train(mpg ~ wt + hp, data = mtcars,
+                       method = "lm",
+                       trControl = u4_TQ3_ctrl)
```

… (4.54)

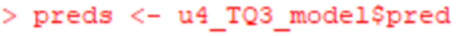
These codes have created all the required information. Root mean square error (RMSE), mean absolute error (MAE) and R-squared can be obtained by simply typing u4_TQ3_model[[4]]$RMSE, u4_TQ3_model[[4]]$MAE and u4_TQ3_model[[4]]$Rsquared on R console and hitting the enter key. Screenshot of the output is shown as follows.



```
> u4_TQ3_model[[4]]$RMSE
[1] 2.715328
> u4_TQ3_model[[4]]$MAE
[1] 2.182071
> u4_TQ3_model[[4]]$Rsquared
[1] 0.8587419
```

… (4.55)

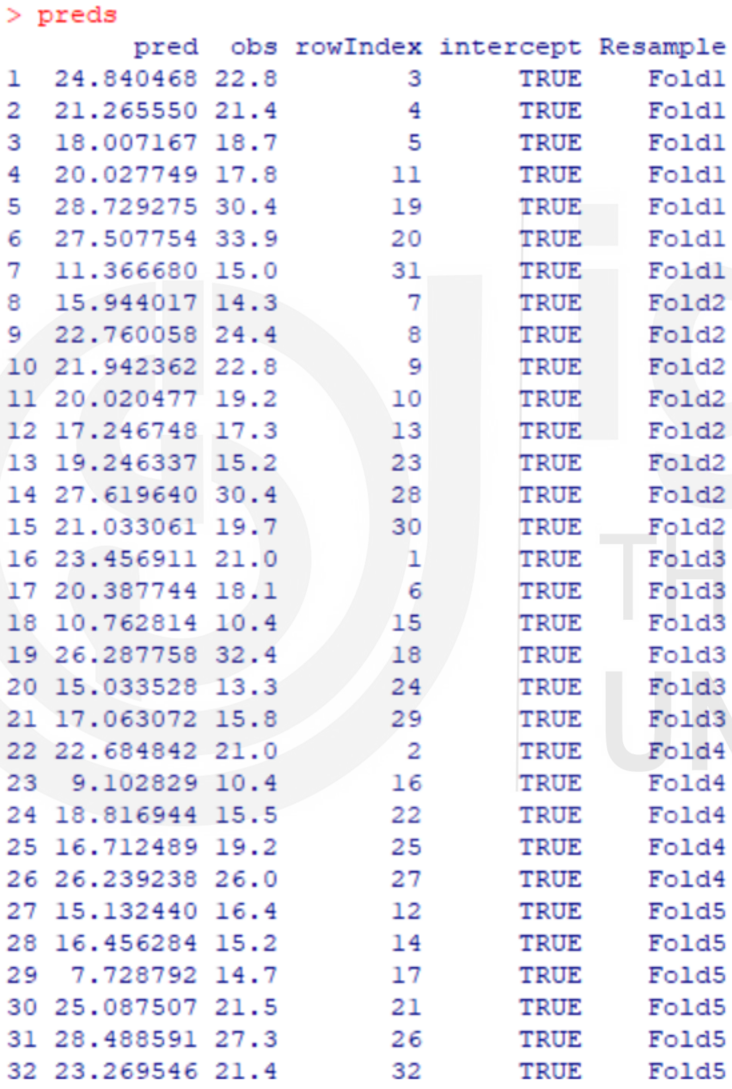
Complete set of predicted values can be stored in the variable preds as follows.



```
> preds <- u4_TQ3_model$pred
```

… (4.56)

Now, all predicted values can be seen by typing preds on R console and hitting enter key. The screenshot of the output is shown as follows.



```
> preds
        pred  obs rowIndex intercept Resample
1  24.840468 22.8        3      TRUE    Fold1
2  21.265550 21.4        4      TRUE    Fold1
3  18.007167 18.7        5      TRUE    Fold1
4  20.027749 17.8       11      TRUE    Fold1
5  28.729275 30.4       19      TRUE    Fold1
6  27.507754 33.9       20      TRUE    Fold1
7  11.366680 15.0       31      TRUE    Fold1
8  15.944017 14.3        7      TRUE    Fold2
9  22.760058 24.4        8      TRUE    Fold2
10 21.942362 22.8        9      TRUE    Fold2
11 20.020477 19.2       10      TRUE    Fold2
12 17.246748 17.3       13      TRUE    Fold2
13 19.246337 15.2       23      TRUE    Fold2
14 27.619640 30.4       28      TRUE    Fold2
15 21.033061 19.7       30      TRUE    Fold2
16 23.456911 21.0        1      TRUE    Fold3
17 20.387744 18.1        6      TRUE    Fold3
18 10.762814 10.4       15      TRUE    Fold3
19 26.287758 32.4       18      TRUE    Fold3
20 15.033528 13.3       24      TRUE    Fold3
21 17.063072 15.8       29      TRUE    Fold3
22 22.684842 21.0        2      TRUE    Fold4
23  9.102829 10.4       16      TRUE    Fold4
24 18.816944 15.5       22      TRUE    Fold4
25 16.712489 19.2       25      TRUE    Fold4
26 26.239238 26.0       27      TRUE    Fold4
27 15.132440 16.4       12      TRUE    Fold5
28 16.456284 15.2       14      TRUE    Fold5
29  7.728792 14.7       17      TRUE    Fold5
30 25.087507 21.5       21      TRUE    Fold5
31 28.488591 27.3       26      TRUE    Fold5
32 23.269546 21.4       32      TRUE    Fold5
```

… (4.57)

ignou
THE PEOPLE'S
UNIVERSITY

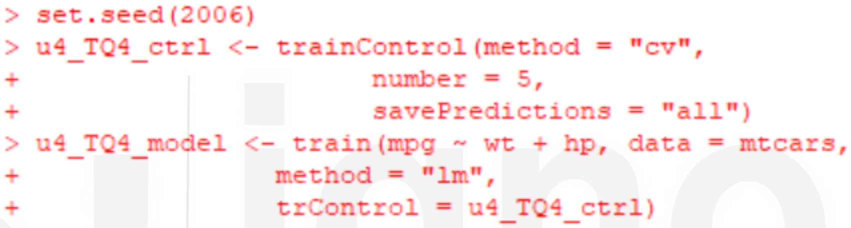
Interpretation of the output of the model:

Root Mean Square Error (RMSE) and **Mean Absolute Error (MAE)** both give average distance of error from 0 but their procedure of removing negative sign of error differ. In RMSE, we square the difference of respective actual values and predicted values of the target variable to remove negative sign, then take average and finally take square root. In MAE, we take absolute value of the difference of respective actual values and predicted values of the target variable to remove negative sign, then take average. … (4.58)

R-Squared: You are familier with the interpretation of R-squared from the course MST-017. Here, its value is 0.8587419 which means all features of the dataset mtcars are able to explain 85.87% of the variation in the target variable mpg. … (4.59)

Remark 1: Note that RMSE in Terminal Question 3 is 2.715328 when we considered only two features wt and hp instead of all features that were considered in SAQ 3. RMSE in SAQ3 was 3.534675. This may be surprising for you. It happens because additional variables introduce noise, multicollinearity and overfitting. Also, simpler model generalises better in cross—validation. So, more variables do not guarantee a better model. You have studied all these details in the course MST-017.

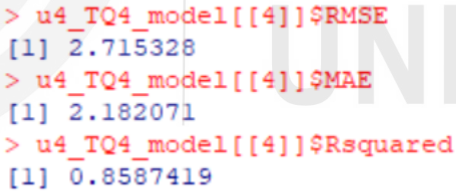
4. As promised in (4.43), we will use trainControl() and train() functions of caret package in R to train lm() machine learning algorithm on mtcars dataset. This can be done as follows.



```
> set.seed(2006)
> u4_TQ4_ctrl <- trainControl(method = "cv",
+                             number = 5,
+                             savePredictions = "all")
> u4_TQ4_model <- train(mpg ~ wt + hp, data = mtcars,
+                       method = "lm",
+                       trControl = u4_TQ4_ctrl)
```

… (4.60)

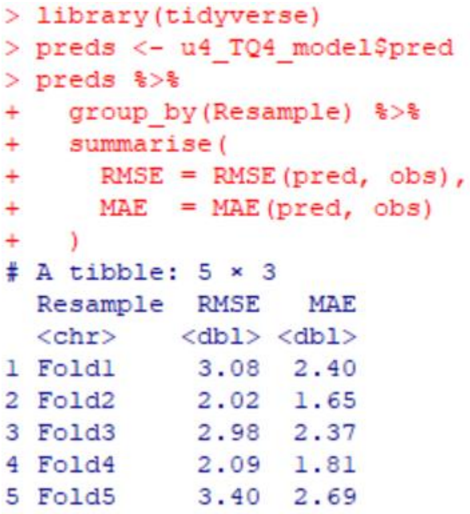
These codes have created all the required information. Root mean square error (RMSE), mean absolute error (MAE) and R-squared can be obtained by simply typing u4_TQ4_model[[4]]$RMSE, u4_TQ4_model[[4]]$MAE and u4_TQ4_model[[4]]$Rsquared on R console and hitting the enter key. Screenshot of the output is shown as follows.



```
> u4_TQ4_model[[4]]$RMSE
[1] 2.715328
> u4_TQ4_model[[4]]$MAE
[1] 2.182071
> u4_TQ4_model[[4]]$Rsquared
[1] 0.8587419
```

… (4.61)

RMSE and MAE for each can be obtained as follows. Screenshot of required R code and its output is shown as follows.

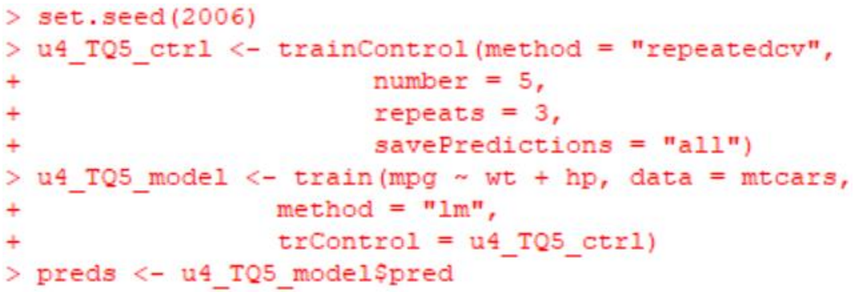


```
> library(tidyverse)
> preds <- u4_TQ4_model$pred
> preds %>%
+   group_by(Resample) %>%
+   summarise(
+     RMSE = RMSE(pred, obs),
+     MAE  = MAE(pred, obs)
+   )
# A tibble: 5 × 3
  Resample  RMSE   MAE
  <chr>    <dbl> <dbl>
1 Fold1     3.08  2.40
2 Fold2     2.02  1.65
3 Fold3     2.98  2.37
4 Fold4     2.09  1.81
5 Fold5     3.40  2.69
```

… (4.62)

ignou
THE PEOPLE'S
UNIVERSITY

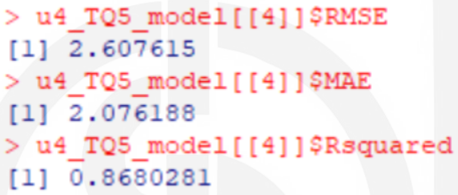
5. As promised in (4.43), we will use trainControl() and train() functions of caret package in R to train lm() machine learning algorithm on mtcars dataset. This can be done as follows.



```
> set.seed(2006)
> u4_TQ5_ctrl <- trainControl(method = "repeatedcv",
+                             number = 5,
+                             repeats = 3,
+                             savePredictions = "all")
> u4_TQ5_model <- train(mpg ~ wt + hp, data = mtcars,
+                       method = "lm",
+                       trControl = u4_TQ5_ctrl)
> preds <- u4_TQ5_model$pred
```

… (4.63)

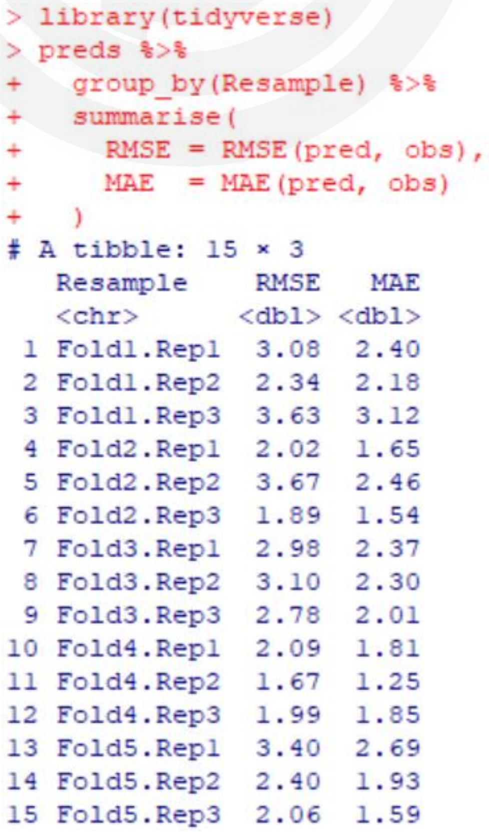
These codes have created all the required information. Root mean square error (RMSE), mean absolute error (MAE) and R-squared can be obtained by simply typing u4_TQ4_model[[4]]$RMSE, u4_TQ4_model[[4]]$MAE and u4_TQ4_model[[4]]$Rsquared on R console and hitting the enter key. Screenshot of the output is shown as follows.



```
> u4_TQ5_model[[4]]$RMSE
[1] 2.607615
> u4_TQ5_model[[4]]$MAE
[1] 2.076188
> u4_TQ5_model[[4]]$Rsquared
[1] 0.8680281
```

… (4.64)

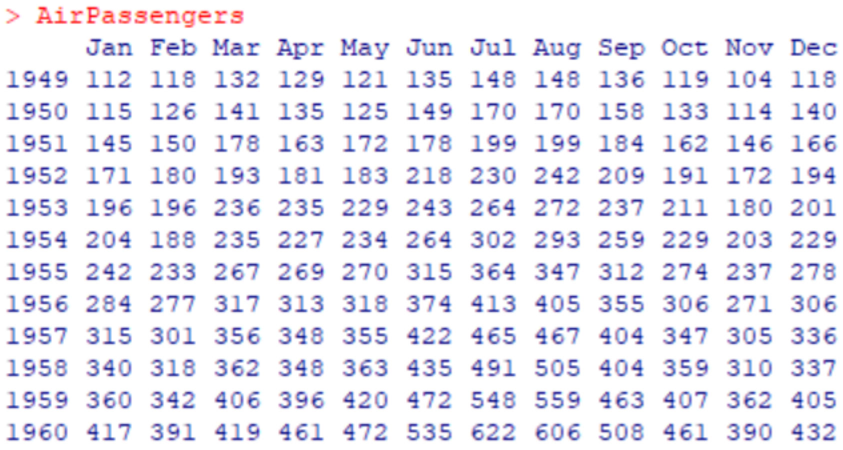
RMSE and MAE for each fold and each repeat can be obtained as follows. Screenshot of required R code and its output is shown as follows.



```
> library(tidyverse)
> preds %>%
+   group_by(Resample) %>%
+   summarise(
+     RMSE = RMSE(pred, obs),
+     MAE  = MAE(pred, obs)
+   )
# A tibble: 15 × 3
   Resample     RMSE   MAE
   <chr>       <dbl> <dbl>
 1 Fold1.Rep1   3.08  2.40
 2 Fold1.Rep2   2.34  2.18
 3 Fold1.Rep3   3.63  3.12
 4 Fold2.Rep1   2.02  1.65
 5 Fold2.Rep2   3.67  2.46
 6 Fold2.Rep3   1.89  1.54
 7 Fold3.Rep1   2.98  2.37
 8 Fold3.Rep2   3.10  2.30
 9 Fold3.Rep3   2.78  2.01
10 Fold4.Rep1   2.09  1.81
11 Fold4.Rep2   1.67  1.25
12 Fold4.Rep3   1.99  1.85
13 Fold5.Rep1   3.40  2.69
14 Fold5.Rep2   2.40  1.93
15 Fold5.Rep3   2.06  1.59
```

… (4.65)

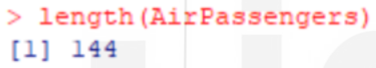
6. Let us first have look at the complete dataset AirPassengers. Screenshot of the output is shown as follows.



```
> AirPassengers
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
```

(4.66)

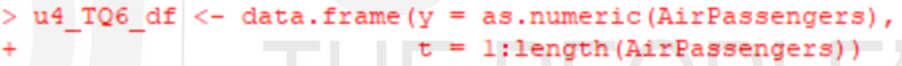
Next, let us see how many observations are there in this dataset. This information can be obtained by applying length() function. Screenshot of the output is shown as follows.



```
> length(AirPassengers)
[1] 144
```

… (4.67)

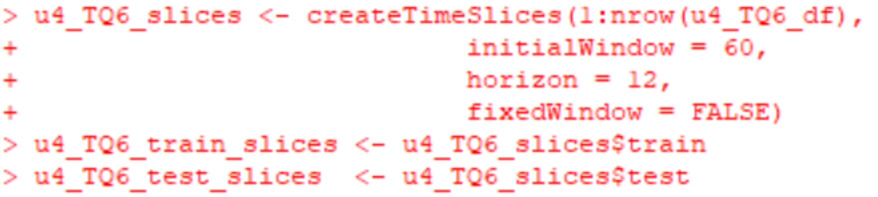
We see that there are 144 observations in this dataset. Now, let us first create data frame u4_TQ6_df having two variables y and t as mentioned in the statement of the question. This can be done as follows.



```
> u4_TQ6_df <- data.frame(y = as.numeric(AirPassengers),
+                         t = 1:length(AirPassengers))
```

… (4.68)

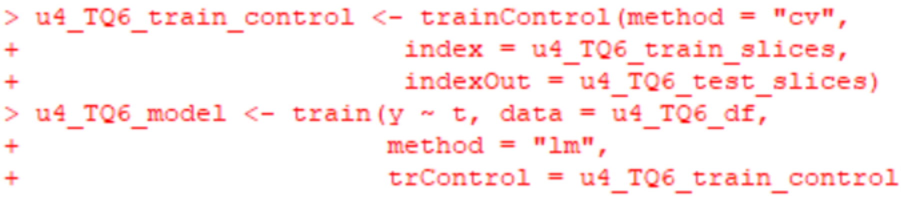
As promised in (4.43), we will use trainControl() and train() functions of caret package in R to train lm() machine learning algorithm on u6_TQ6_df dataset. But as given in the statement of the question before that first, we have to set initial training window (initialWindow) to 60 months and the testing window (horizon) to 12 months. This can be done as follows.



```
> u4_TQ6_slices <- createTimeSlices(1:nrow(u4_TQ6_df),
+                                  initialWindow = 60,
+                                  horizon = 12,
+                                  fixedWindow = FALSE)
> u4_TQ6_train_slices <- u4_TQ6_slices$train
> u4_TQ6_test_slices  <- u4_TQ6_slices$test
```

… (4.69)

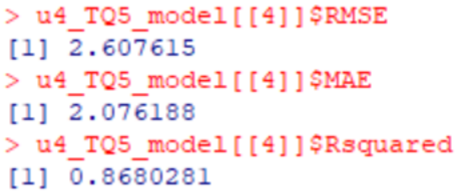
Now, we will use trainControl() and train() functions of caret package in R to train lm() machine learning algorithm on u6_TQ6_df dataset.



```
> u4_TQ6_train_control <- trainControl(method = "cv",
+                              index = u4_TQ6_train_slices,
+                              indexOut = u4_TQ6_test_slices)
> u4_TQ6_model <- train(y ~ t, data = u4_TQ6_df,
+                       method = "lm",
+                       trControl = u4_TQ6_train_control)
```

… (4.70)

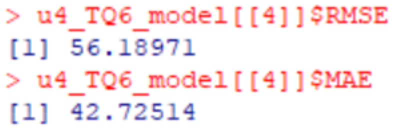
These codes have created all the required information. Root mean square error (RMSE) and mean absolute error (MAE) can be obtained by simply typing u4_TQ6_model[[4]]$RMSE, u4_TQ6_model[[4]]$MAE on R console and hitting the enter key. Screenshot of the output is shown as follows.



```
> u4_TQ5_model[[4]]$RMSE
[1] 2.607615
> u4_TQ5_model[[4]]$MAE
[1] 2.076188
> u4_TQ5_model[[4]]$Rsquared
[1] 0.8680281
```

… (4.71)

RMSE and MAE for each fold and each repeat can be obtained as follows. Screenshot of required R code and its output is shown as follows.



```
> u4_TQ6_model[[4]]$RMSE
[1] 56.18971
> u4_TQ6_model[[4]]$MAE
[1] 42.72514
```

… (4.72)

ignou
THE PEOPLE'S
UNIVERSITY